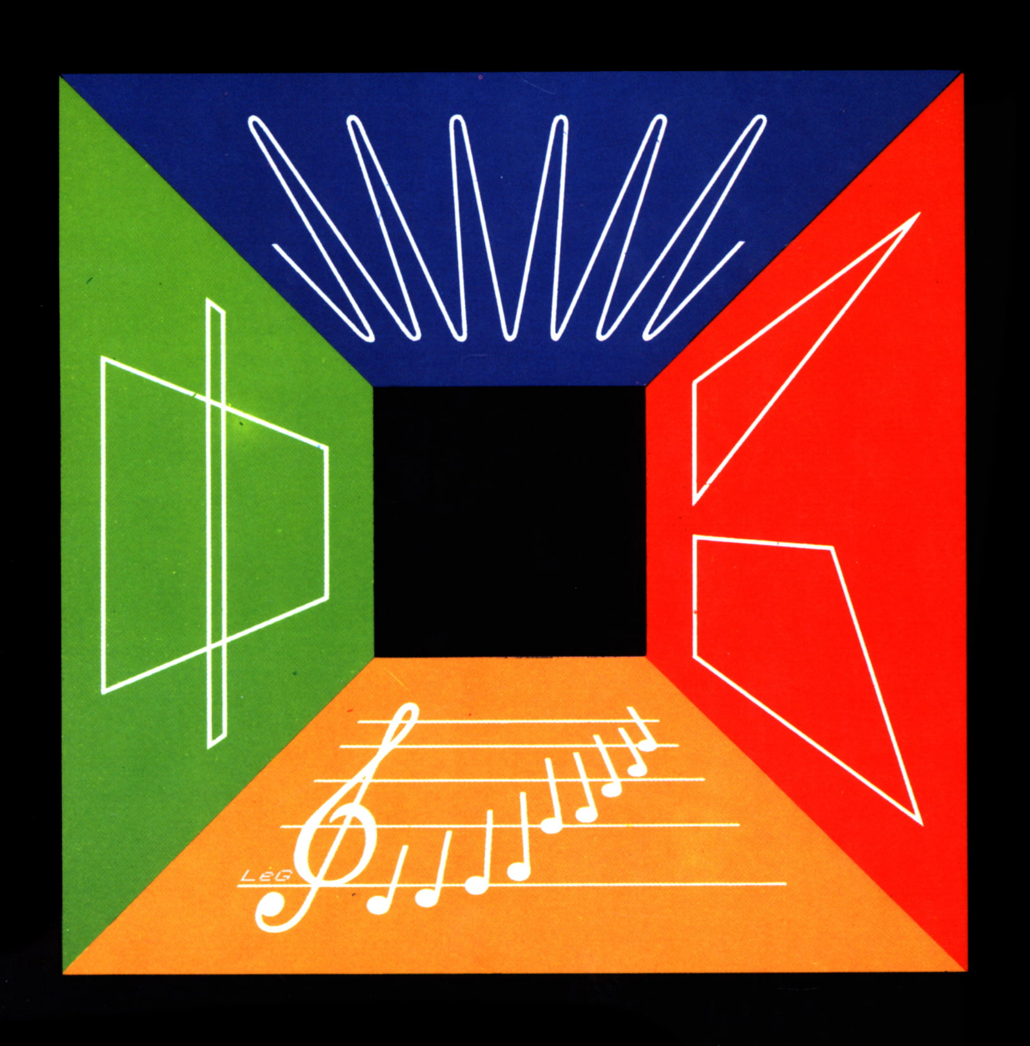
GRAFICOS Y SONIDO EN EL SPECTRUM



STEVE MONEY

Gráficos y sonido en el Spectrum

Gráficos y sonido en el Spectrum

Steve Money

DIAZ DE SANTOS, S.A.
MADRID-ESPAÑA

Titulo original: "Spectrum. Grafics and sound".

ISBN: 0-246-12192-0 (edición en lengua inglesa).
ISBN: 84-86251-14-1 (edición en lengua española).

Depósito legal: M. 5.346-1985

EDITA: DIAZ DE SANTOS, S.A. 1.ª edición, 1985
C/ Juan Bravo, 3A - 28006-MADRID
PRODUCCION: A.S.E.L., S.A.
C/ Clara del Rey, 41 - Teléf. 416 62 47
28002-MADRID
Traductor: Blanca Arbizu Duralde
Revisado por: AULA DE INFORMATICA APLICADA (AIA)
C/ Conde de Peñalver, 38 - 28006-MADRID

Impreso en Lavel. Los Llanos, nave 6. Humanes (Madrid)

Contenido

Prefacio

El SPECTRUM es uno de los ordenadores personales más populares entre los que tienen posibilidades de visualización de gráficos en color. Dado que dispone de capacidad de alta resolución para la realización de gráficos, puede producir sobre la pantalla diseños de muy buena calidad. Esta cualidad es muy útil en la visualización de gráficos y diagramas y, evidentemente, es un ingrediente esencial en muchos de los programas de juegos que se han escrito para el Spectrum. En este libro veremos las técnicas necesarias para producir dibujos y diagramas sobre la pantalla.

El primer capítulo explica, en términos sencillos, cómo se crea la imagen en la pantalla, y estudia algunas de las características originales que utilizaremos en el Spectrum.

En el Capítulo 2, exploraremos las posibilidades que ofrecen los símbolos gráficos de mosaico, que vienen incluidos en el conjunto de símbolos standard del Spectrum. Estos símbolos se imprimen en la pantalla del mismo modo que los caracteres de texto. A diferencia de otros ordenadores, el Spectrum no va provisto de instrucciones de trazado o de dibujo para usar con los símbolos gráficos de mosaico, pero en el Capítulo 2 veremos métodos de manejo de estos símbolos para dibujar líneas, e incluso para proporcionar un programa sencillo de bosquejo.

Quizás la cualidad más atractiva que presenta el Spectrum es la posibilidad de utilizar alta resolución en la creación de gráficos en color. En los Capítulos 3 y 4, estudiaremos las técnicas apropiadas para el diseño de varios tipos de figuras, como polígonos y círculos. Desafortunadamente, el método que utiliza el Spectrum para almacenar la información en color impone ciertas limitaciones en el uso del color para el modo de alta resolución; sin embargo, con algo de cuidado, se pueden obtener muy buenos resultados.

El Spectrum proporciona al usuario posibilidades de diseñar símbolos a medida y, en el Capítulo 5, examinaremos algunas de estas opciones. Resulta muy fácil producir en la pantalla una gran

variedad de versiones de los símbolos. Con este fin, se suministran los correspondientes programas. También es posible obtener más tonalidades de color con los ocho colores básicos que proporcionan los comandos INK y PAPER del Spectrum.

En el Capítulo 6, estudiaremos las técnicas de obtención de estas tonalidades.

Una de las aplicaciones de las más prácticas del Spectrum es la visualización de gráficos y diagramas de diversos tipos. En el Capítulo 7 veremos algunas técnicas relacionadas con el dibujo de gráficos, diagramas de barras, diagramas circulares, etc.

En cuanto a los juegos, evidentemente el movimiento es un factor importante. En el Capítulo 8, estudiaremos las técnicas y principios básicos que podemos utilizar en la animación de objetos sobre la pantalla de visualización. Por desgracia, el lenguaje BASIC del Spectrum es relativamente lento, y para una acción muy rápida necesitaremos acudir al código máquina, tema que rebasa los objetivos de este libro.

En la pantalla del Spectrum también se pueden producir imágenes pseudotridimensionales. Las podemos realizar utilizando diagramas y gráficos con tres ejes, y técnicas de perspectiva que proporcionan visiones más reales de escenas y objetos. Incluimos un programa que permite la visión de un objeto desde cualquier ángulo.

Finalmente, llegamos a la generación de música y sonido. A este respecto, el Spectrum es muy limitado, ya que sólo lleva una sencilla instrucción de producción de sonido (BEEP). Sin embargo, con el Spectrum pueden crearse canciones, e incluso podemos llegar a convertirlo en un instrumento musical sencillo. Esto lo veremos en el Capítulo 10.

La pretensión de este libro ha sido explicar algunas de las técnicas de utilización de las cualidades del Spectrum para la creación de gráficos y sonido. Sin embargo, una de las mayores diversiones que nos proporcionan los ordenadores es la exploración de nuevas ideas, y este libro tiene la esperanza de ofrecerle una guía que le permita explorar las posibilidades que ofrece el ordenador Spectrum.

Steve Money

Capítulo 1

Introducción a los gráficos

Una de las características más atractivas de casi todos los ordenadores personales modernos es su habilidad en la producción y visualización de gráficos de gran colorido. Esta característica permite programar al ordenador de forma que visualice, sobre la pantalla de televisión, esquemas, diseños, gráficos, diagramas y dibujos con animación.

Todos los ordenadores personales populares tienen sistemas para producir gráficos de algún tipo, y algunos de ellos tienen, además, la atracción del color. La calidad de las visualizaciones que se producen sobre la pantalla depende de un término llamado *resolución* gráfica, que da la medida de la finura de los detalles en la visualización de un dibujo. La resolución se mide como el número de puntos que se pueden controlar individualmente en las dos direcciones de la pantalla. Así, una resolución de 256 x 176 significa que existen 256 puntos en una dirección de la pantalla y 176 en la otra, cada uno de ellos controlable individualmente (encendido-si, apagado-no). En general, cuanto más alta es la resolución, mejor es la visualización del dibujo sobre la pantalla.

Uno de los problemas que presenta la alta resolución en la visualización de gráficos es que suele requerir gran cantidad de memoria. Si la visualización es en color, se necesitará aún más memoria para el almacenamiento de la información del color de cada uno de los puntos individuales de la pantalla. Con el fin de ahorrar memoria, algunos ordenadores limitan el número de colores que pueden visualizarse en un momento dado. En el Spectrum se utiliza una técnica que, con algunas limitaciones, permite el uso de ocho colores diferentes simultáneamente, sin que ello suponga una reducción muy grande de la memoria disponible.

Una de las utilizaciones más atractivas de los ordenadores personales es la emulación de los juegos de video que se encuentran normalmente entre la gama de los juegos de recreo. Juegos típicos de

estas características son: los Invasores, Asteroides, Defensores, etc. Este tipo de juegos populares se basa en gran medida en la visualización de gráficos de brillantes colores y en movimiento rápido sobre la pantalla. Los juegos actuales utilizan microprocesadores similares a los utilizados en los ordenadores personales. Es más, estas máquinas de juegos son, frecuentemente, computadoras potentes, pero están dedicadas específicamente a juegos de video.

Estos juegos pueden presentarse en una televisión convencional, utilizando una consola de video-juegos, por ejemplo, tipo Atari.

Estas unidades utilizan también un microprocesador, pero, a diferencia de las otras máquinas, pueden generar una gama de juegos muy amplia. En este tipo de máquinas, el programa de ordenador, para cada juego diferente, se suministra en un cartucho que se introduce en la consola.

Los ordenadores utilizados como máquinas de juegos

Un ordenador personal, como, por ejemplo, el Spectrum puede utilizarse también como máquina de juegos, con tal que disponga de una buena capacidad de visualización de gráficos. En este caso, el ordenador está programado para visualizar un dibujo de la situación del juego, con movimiento, en tiempo real, y con piedras, naves espaciales o extraterrestres moviéndose rápidamente por la pantalla de televisión. El sonido y el color son también factores importantes en este tipo de juegos, y, de hecho, el Spectrum cumple muy bien estas condiciones.

Una de las mayores ventajas del ordenador personal, comparado con las máquinas de video-juegos, es que con el ordenador usted puede inventar sus propios juegos, y programarlos utilizando o bien el BASIC, o bien el código de máquina. Una vez haya escrito los programas, puede guardarlos en una cassette para su uso posterior.

Además de los juegos descritos, existe otra clase de juegos a su alcance para utilizar con su ordenador. Estos forman toda una gama que va desde la sencillez de adivinanzas con números, hasta juegos muy complicados, como puede ser el ajedrez. De este tipo de juegos, los más populares son los juegos de aventuras. Variaciones sobre este tema pueden ser los juegos de laberintos, o juegos en los que el jugador participa en alguna forma, como en las Mazmorras y los Dragones. En los juegos de aventuras, el jugador explora mundos misteriosos, moviéndose de un lugar a otro en busca de tesoros, luchando contra monstruos, y resolviendo enigmas. El ordenador

actúa como su doble, le dice lo que puede ver, y desarrolla acciones, como, por ejemplo, recoger objetos. En el caso de las Mazmorras y los Dragones, el jugador puede ir adoptando diferentes papeles según va explorando.

En muchos juegos de aventuras, las posiciones se describen mediante un texto escrito en la pantalla. A este tipo de juego se le denomina de aventuras "con texto". Los juegos de aventuras se realizan mucho más si en la pantalla de la televisión puede verse la posición que en cada momento tiene el explorador, en lugar de ofrecer únicamente una descripción escrita. También se pueden mostrar dibujos de objetos o tesoros que se encuentran en la posición del explorador en ese momento. En algunas versiones, al explorador se le permite examinar con más detalle un objeto seleccionado. En este caso, la imagen del objeto se amplía hasta llenar la pantalla.

El laberinto es la variante más sencilla de los juegos de aventuras. En este caso, el explorador se encuentra en el centro de un laberinto y tiene que encontrar el camino de la salida. Las versiones más sencillas describen el camino a tomar con palabras, pero las versiones más atractivas muestran un dibujo de lo que el explorador puede ver cuando mira el camino sobre el que se encuentra. Si dibujamos una vista en perspectiva a lo largo del pasillo, en el laberinto, los giros pueden verse con un realismo fantástico. A medida que el jugador se mueve, el dibujo sobre la pantalla se modifica para mostrar la perspectiva que se observa desde la nueva posición. Lo más avanzado en este tipo de juegos es un laberinto tridimensional en el que el explorador puede moverse no sólo hacia adelante o hacia atrás, a izquierdas o a derechas, sino también hacia arriba y abajo en varios niveles del laberinto.

Otras utilizaciones de los gráficos

Evidentemente, los juegos son un pasatiempo muy divertido, pero la mayoría de los poseedores de un ordenador, así como los usuarios, utilizan también las computadoras para actividades más serias. En multitud de aplicaciones, un gráfico o un diagrama puede ser mucho más efectivo que una lista de números, para mostrarnos el progreso de un negocio, o cualquier otra actividad. Los gráficos y diagramas son también importantes en experimentación científica, cuando deseemos visualizar los resultados de un experimento científico o la información obtenida de una votación o de una encuesta. Todos estos diferentes resultados, como diagramas de barras, circu-

lares y dibujos gráficos convencionales, pueden visualizarse en el Spectrum con mucha efectividad.

Otra aplicación diferente es el dibujo de gráficos sobre el ordenador, utilizando un sistema de visualización de alta resolución. Quizás podamos producir dibujos técnicos como planos, diagramas de circuitos electrónicos o ilustraciones de libros. Muchas oficinas de ingeniería de diseño utilizan técnicas de ayuda por computadora en sus dibujos y diseños, mediante un conjunto de programas denominado DISEÑO ASISTIDO POR COMPUTADORA (computer aided design-CAD). También nosotros podemos utilizar técnicas similares para mostrar el conjunto de dispositivos utilizados en un experimento físico o químico, y sus resultados. Esto puede ser muy útil en el caso de que el equipo necesario para desarrollarlo sea muy caro. En este caso, el ordenador simula el experimento que estamos realizando, pudiéndose visualizar los diferentes estudios en el proceso del experimento, dando lecturas de, por ejemplo, la temperatura, presión, etc.

Evidentemente, los ordenadores pueden utilizarse también con fines puramente artísticos. En este campo, el ordenador puede producir formas muy bonitas, con el manejo del color, e incluso puede generar vistas con perspectiva. Si se utilizan técnicas de visualización de objetos con volumen, podemos dibujar un objeto y luego verlo desde direcciones diferentes. Podemos también modificar la forma resultante mientras la vemos, hasta obtener el resultado apetecido.

La pantalla de video

Los ordenadores personales modernos, incluido el Spectrum, utilizan la pantalla de televisión para visualizar la salida de un programa de ordenador. Puede usarse, por tanto, el receptor de televisión doméstico, o bien un monitor de televisión especial diseñado para utilizarse como unidad de visualización de la computadora (VDU). El monitor es muy útil en las visualizaciones en color, y dará dibujos más claros y estables, ya que las técnicas electrónicas utilizadas en la transmisión y recepción por televisión tienden a degradar los dibujos gráficos. Por ello, puede resultar útil comprobar el aspecto que en este momento presentan nuestros diseños en la pantalla.

Aunque la imagen en la pantalla de televisión nos parece un dibujo completo y estático, está formada por un único punto que se mueve y barre muy rápidamente la pantalla en series de líneas hori-

zontales. A medida que el punto se mueve en la pantalla, atravesándola de izquierda a derecha, también se mueve hacia abajo mucho más lentamente, de modo que las sucesivas líneas se van produciendo justo debajo de la última línea trazada. Una vez alcanzado el ángulo inferior derecho de la pantalla, el punto se mueve con toda rapidez al ángulo superior izquierdo, y el proceso se vuelve a repetir. La operación de exploración de la pantalla se realiza continuamente, y el punto barre la pantalla en 1/50 segundos. El número total de líneas de barrido es 625.

Cuando miramos la pantalla, nuestros ojos son incapaces de ver el punto que se mueve, porque va a una velocidad muy rápida y el ojo no responde en la visión a velocidades superiores a 1/20 segundos. Además, el material de la pantallla está diseñado de tal modo que con el movimiento del punto se ilumina, pero no se apaga inmediatamente, sino que permanece iluminado aproximadamente 1/20 segundos. Como resultado de todo ello, vemos la imagen completa como si estuviera presente en la pantalla continuamente. Las líneas se encuentran entrelazadas para reducir el parpadeo, y el punto barre la pantalla, pasando, por ejemplo, por las líneas impares, y en el próximo barrido recorre los huecos pasando por las pares. Esta acción produce un parpadeo de 50 por segundo, invisible para el ojo humano, mientras que el trazado de todas las líneas cada 1/25 segundos produciría un parpadeo detectable, que podría distraer al espectador Podemos variar la brillantez del punto en su barrido por la pantalla, creando un patrón de luminosidad y oscuridad con los puntos, que es el que forma la imagen que vemos. La televisión en color tiene una pantalla especial, cuyos puntos, colocados en pequeños grupos, de tres en tres, pueden dar luz azul, roja o verde. En el interior del tubo, existen tres haces de barrido diferentes, que seleccionan el rojo, el verde y el azul, respectivamente. Utilizando diferentes combinaciones de luz roja, verde o azul en cada conjunto de tres puntos, se puede iluminar cualquier punto de la pantalla con el color deseado.

Visualización de textos

En el momento en el que usted conecte su Spectrum, aparecerá una pantalla blanca con texto en negro escrito sobre ella. Antes de estudiar la visualización de gráficos, puede ser útil inspeccionar la forma en la que un ordenador personal típico genera símbolos de texto sobre la pantalla de televisión.

Para la visualización del texto, la pantalla está dividida en un conjunto de espacios rectangulares, cada uno de los cuales se denomina espacio de caracteres o espacio de símbolos. En cada uno de estos espacios pueden visualizarse una letra, un número o cualquier otro símbolo. Para la presentación del texto, el ordenador Spectrum divide la pantalla en 24 líneas horizontales, con 32 símbolos por cada línea. Como veremos más adelante, se reservan dos líneas para uso de la computadora, dejando, por tanto, 22 líneas para la visualización del texto.

Si examinamos de cerca los símbolos, se verá que cada letra está formada por un conjunto de puntos individuales. El Spectrum utiliza un conjunto de ocho hileras de ocho puntos cada una de ellas, en el interior del espacio del carácter. La forma del símbolo del texto se consigue iluminando algunos de los puntos y dejando en la oscuridad los demás. Un ejemplo de esta técnica puede verse en la figura 1.1. Una vez seleccionados los patrones a visualizar para cada símbolo, puede crearse sobre la pantalla la imagen completa.

El Spectrum imprime su texto en letras negras, que aparecen contrastando con el fondo blanco. En este caso, los puntos que generan la forma de las letras no se iluminan.

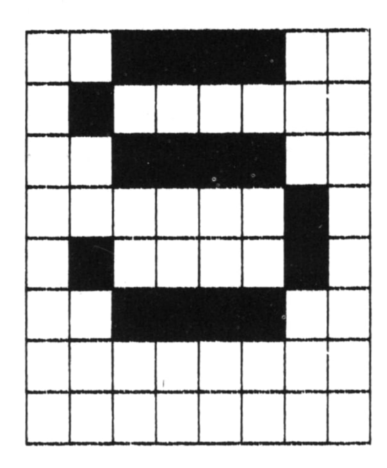


Fig. 1.1. Matriz de puntos usada como patrón para presentar en pantalla los símbolos de texto.

Dado que la imagen completa se traza cincuenta veces por segundo, necesitaremos tener guardados, en alguna parte de la memoria, los datos que representen la totalidad del texto que se desea visualizar. De esta forma, el sistema lógico de visualización

puede utilizar los patrones de puntos según se vayan necesitando, para producir la presentación del texto en la pantalla. La estructura normal del sistema de visualización se muestra de forma simplificada en la figura 1.2. Una parte de la memoria, llamada normalmente memoria de visualización, es la que se utiliza para guardar los datos que definen los símbolos del texto que se visualiza. Normalmente, se almacena un código de 8 bits para cada símbolo de visualización. Cada posición de la memoria también contiene 8 bits de datos, de modo que cada símbolo de texto ocupa un lugar en la memoria de visualización.

Los patrones de puntos que configuran los diferentes símbolos, están guardados en un dispositivo de memoria especial llamado *GENERADOR DE CARACTERES*. Este generador es básicamente un dispositivo de memoria, pero, a diferencia de la memoria normal de un ordenador, los patrones de datos están escritos permanentemente desde el momento de su creación, y no pueden borrarse ni cambiarse.

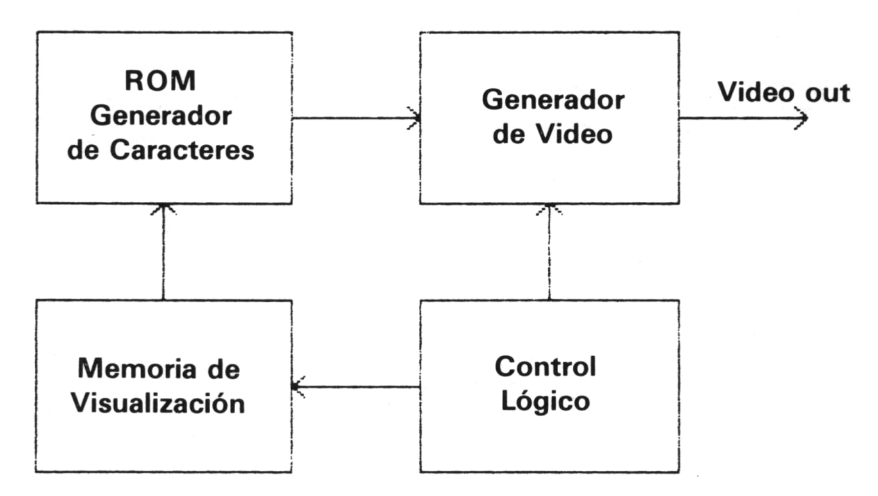


Fig. 1.2. Diagrama de bloques básico, del sistema de presentación de textos en la pantalla de un ordenador.

Este tipo de memoria se llama Read Only Memory (Memoria de Sólo Lectura) o ROM. Cuando se le facilita al generador de caracteres el código del carácter, éste selecciona el patrón adecuado con los puntos, y, a partir de ello, pueden leerse los datos que representan dichos puntos, y generar las señales de video que producirán los puntos en la pantalla.

Mientras se va trazando el dibujo, los datos de cada espacio de carácter se leen de la memoria de visualización y se pide al generador de caracteres el patrón de puntos adecuado. Los patrones de puntos se convierten entonces en una señal de video similar a la que produce imágenes en la televisión de nuestro hogares. Una vez alimentada la televisión con todas estas señales, se produce la visualización del texto sobre la pantalla. La lógica y la electrónica necesarias para estos procesos, son muy complejas, y en muchos ordenadores se utiliza un circuito integrado especial o "chip", para el control de la generación de la imagen en la pantalla.

El Spectrum difiere bastante del resto de los ordenadores en el modo en que produce la visualización del texto. Como las demás máquinas, tiene una sección de la memoria que contiene permanentemente los patrones de puntos que forman el conjunto de los símbolos que se pueden visualizar. Sin embargo, en lugar de almacenar el código de un símbolo de texto en su memoria de visualización, almacena el patrón de puntos necesario en ese momento. Así pues, cuando se llama a un carácter para su visualización, su patrón de puntos se copia del generador de caracteres en la memoria de visualización. Como cada patrón de símbolos contiene ocho hileras de puntos, la memoria de visualización de la pantalla es mucho mayor que si sólo almacenara el código de carácter. De hecho, la memoria que utiliza el Spectrum es ocho veces mayor que lo que sería si utilizara técnicas convencionales. Como veremos, esta técnica tiene ventajas, especialmente cuando se desee mezclar en la pantalla gráficos de alta resolución junto con símbolos.

Gráfico de mosaicos

En los primeros tiempos de los ordenadores, los gráficos y dibujos se producían frecuentemente utilizando símbolos de texto con los que se formaban los detalles del dibujo. Algunas letras son más brillantes que otras por su misma forma, y, por tanto, eligiendo cuidadosamente una letra para un punto determinado, podemos obtener una imagen clara u obscura. Si la página de texto se coloca a suficiente distancia de los ojos del observador, éste no será capaz de distinguir las letras individuales, y verá el dibujo completo. Esta técnica no es muy adecuada para el Spectrum, porque no tiene suficientes símbolos en la pantalla para obtener un dibujo correcto.

La utilización de símbolos no es muy efectiva, y muchos de los ordenadores personales tienen, entre el conjunto de sus caracteres,

símbolos especiales que permiten el dibujo sobre la pantalla. Estos símbolos especiales pueden ser un segmento de una línea vertical, horizontal o diagonal dentro del espacio del carácter. Otro tipo de símbolos puede visualizar esquinas o uniones en T, e incluso líneas curvas. Colocando cuidadosamente estos símbolos especiales, se pueden producir dibujos e imágenes muy detalladas. Otros símbolos especiales disponibles pueden incluir, por ejemplo, símbolos de naipes o incluso dibujos de piezas de ajedrez. Algunos ordenadores permiten que el usuario cree símbolos a medida, almacenando el patrón de puntos en la memoria central y no en el generador de caracteres ROM.

Esta técnica de utilización de caracteres gráficos especiales para crear una imagen, es restringida si no se utiliza una selección amplísima de símbolos especiales. El Spectrum no va provisto de símbolos especiales standard, pero existe la posibilidad de que el usuario pueda crear símbolos especiales con un diseño propio.

Como alternativa, contamos con otra posibilidad más flexible para producir visualizaciones de gráficos de resolución media a baja. El Spectrum utiliza los llamados símbolos gráficos de *MOSAICO*, que son símbolos gráficos similares a los utilizados en la producción de gráficos en los teletipos y servicios de visualización de datos. Una vez más, se utiliza un conjunto especial de símbolos gráficos, pero, en este caso, el espacio del carácter está dividido en un patrón formado por cuatro bloques pequeños. Cada uno de los elementos del bloque puede iluminarse o permanecer oscuro para formar un patrón basto en el interior del espacio del carácter. La figura 1.3. muestra ejemplos típicos de símbolos gráficos de mosaico. Los patrones de bloque de los símbolos de mosaico pueden utilizarse para formar un dibujo de forma muy similar a como se utilizan los gráficos lineales. Esta técnica ofrece una imagen más basta que si se hubiera creado utilizando gráficos lineales, pero es más flexible.

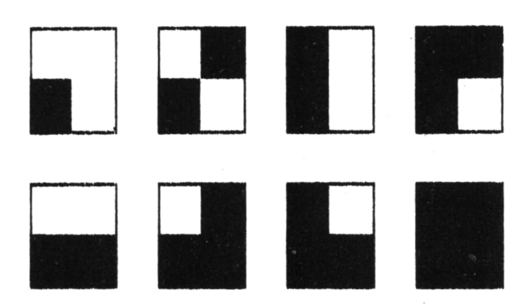


Fig. 1.3. Símbolos de mosaico típicos utilizados en el Spectrum.

En el Spectrum, cada espacio de un símbolo está dividido en cuatro bloques menores. Cada uno de estos bloques puede ser blanco o negro. Existen, pues, dieciséis patrones de bloque que pueden producirse en el interior del espacio de un símbolo. Eligiendo cuidadosamente los patrones de bloque adecuados, podemos producir un dibujo en la pantalla. De este modo tenemos una extraña forma de imagen de televisión. En el Spectrum, este tipo de gráficos proporciona una resolución de 64 puntos en el sentido horizontal, por 44 en sentido vertical.

Aunque el Spectrum normalmente produce los símbolos del texto en blanco y negro, el texto y el mosaico de símbolos gráficos se pueden visualizar también en ocho colores diferentes, siendo, asimismo, posible elegir el color de fondo de cada espacio de símbolo entre cualquiera de los ocho colores anteriores. De este modo, utilizando este tipo de visualización de gráficos, se pueden producir dibujos con mucho colorido. Para hacerse una pequeña idea de la resolución y capacidad de color de estos gráficos de mosaico en la visualización, pruebe a ejecutar el programa listado en la figura 1.4.

```
100 REM Demostracion de graficos de mosaico.
110 LET x=15
120 LET y=10
130 FOR j=1 TO 15
140 INK INT (RND*7)
150 LET k=j
160 IF k>10 THEN LET k=10
170 FOR i=1 TO j
180 PRINT AT y+k,x+i;CHR$ 130;
190 PRINT AT y+k,x-i;CHR$ 129;
200 PRINT AT y-k,x-i;CHR$ 132;
210 PRINT AT y-k,x+i;CHR$ 136;
220 NEXT i
230 FOR i=1 TO k
240 PRINT AT y+i,x+j;CHR$ 130;
250 PRINT AT y+i,x-j;CHR$ 129;
260 PRINT AT y-i,x-j;CHR$ 132;
270 PRINT AT y-i,x+j;CHR$ 136;
280 NEXT i
290 NEXT j
300 FOR n=1 TO 500
310 LET i=INT (RND*16)
320 LET j=INT (RND*10)
330 INK INT (RND*7)
340 PRINT AT y+j,x+i;CHR$ 130;
350 PRINT AT y+j,x-i;CHR$ 129;
360 PRINT AT y-j,x-i;CHR$ 132;
370 PRINT AT y-j,x+i;CHR$ 136;
380 NEXT n
```

Fig. 1.4. Programa-demostración de la utilización de los gráficos de mosaico en el Spectrum.

Como veremos en el siguiente capítulo, la utilización de los símbolos gráficos de mosaico puede resultar muy compleja, para el dibujo de líneas o incluso para colocar un punto en una posición específica. El Spectrum puede programarse eligiendo bloques individuales del mosaico, e iluminando o no estos bloques. Esta técnica presenta, sin embargo, algunas limitaciones en el uso de los colores, pero, de todos modos, se pueden obtener visualizaciones muy atractivas.

Una de las ventajas que presentan estos símbolos gráficos de mosaico es que pueden imprimirse en pantalla de igual modo que si fueran símbolos de texto, y resulta muy sencillo mezclar ambos, texto y gráficos, en el sistema de visualización.

Caracteres especiales y gráficos

Una de las técnicas de los ordenadores para la producción de gráficos, consiste en utilizar una serie de símbolos gráficos de que van provistos, como, por ejemplo, líneas horizontales, verticales y diagonales. También pueden ir dotados de líneas curvas e incluso símbolos especiales, como los palos de la baraja de cartas. Si se seleccionan y se colocan con cuidado estos símbolos especiales en la pantalla, se pueden obtener mejores gráficos que los producidos con los símbolos especiales de mosaico.

El Spectrum tiene la ventaja de poder producir estos símbolos a medida. Los patrones de puntos de estos símbolos especiales pueden colocarse en una zona reservada de la memoria central del ordenador, y retenerse allí mientras el ordenador está conectado. A diferencia de los símbolos de texto normales, estos patrones se perderán en el momento en que desconectemos el ordenador. Los patrones de puntos se transfieren a la memoria principal de pantalla, del mismo modo que el texto normal y que los símbolos gráficos de mosaico.

Algunas veces se necesita un patrón de gráficos mayor, con una matriz de puntos de 16 x 16. Este puede obtenerse a partir de un grupo de símbolos de tamaño standard.

Gráficos de alta resolución; Pixels

Como ya hemos visto anteriormente, los símbolos de texto de la pantalla también se han realizado encendiendo selectivamente ciertos puntos del interior del espacio del símbolo. Sin embargo, los

patrones de puntos disponibles están ya fijados, ya que están gobernados por los conjuntos de patrones de puntos, ubicados en el generador de caracteres.

Suponga que disponemos de un modo de visualización alternativo con el que podemos controlar los estados de los puntos individuales de cada espacio de carácter en la pantalla. En el Spectrum, cada símbolo de texto está formado por 8 filas de puntos. Con 32 espacios de caracteres en sentido horizontal en la pantalla, esto nos da 32 x 8, o sea, un total de 256 puntos en la pantalla para la resolución horizontal o resolución X. Aunque de hecho en la pantalla existen 24 filas de texto, las dos últimas están reservadas para los mensajes del ordenador, por lo que sólo disponemos de 22 filas de visualización. Como el espacio del carácter tiene una altura de 8 puntos y tenemos 22 filas de texto, en total habrá 8 x 22 o un total de 176 puntos, en el sentido vertical en la pantalla.

```
100 REM Programa de demostracion
105 REM de graficos de alta resolucion
110 LET x=0
120 LET y=0
125 REM dibuja lineas verticales
130 FOR w=1 TO 22
140 PLOT x,y
150 DRAW 0,175
160 LET x=x+w
170 NEXT w
180 LET x=0
185 REM dibuja lineas horizontales
190 FOR h=1 TO 19
200 PLOT x,y
210 DRAW 255,0
220 LET y=y+h
230 NEXT h
235 REM dibuja lineas diagonales
240 PLOT 0,0
250 DRAW 255,175
260 PLOT 0,175
270 DRAW 255,-175
275 REM dibuja los bordes
280 PLOT 0,0
290 DRAW 255,0
300 DRAW 0,175
310 DRAW -255,0
320 DRAW 0,-175
```

Fig. 1.5. Programa demostrativo de gráficos de alta resolución obtenidos con el Spectrum.

Los puntos individuales de cada espacio de carácter se llaman, generalmente, pixels (picture elements-elementos de dibujo), y el

modo gráfico que representan se llama modo gráfico de pixels, o de *ALTA RESOLUCION*. La figura 1.6. nos da un ejemplo del tipo de visualización que se puede producir utilizando la capacidad de producción del Spectrum para los gráficos de alta resolución. De hecho, muchas de las ilustraciones que se muestran en este libro han sido generadas con el Spectrum, y luego impresas en una impresora de matriz de puntos.

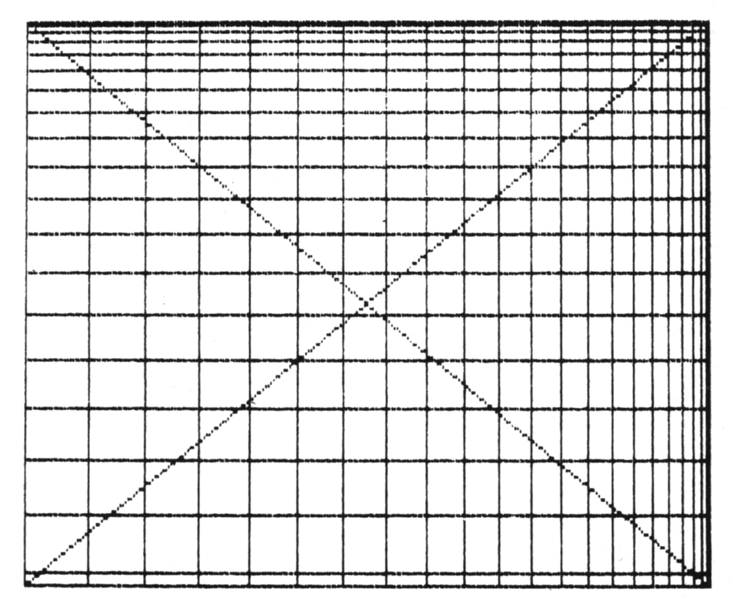


Fig. 1.6. Visualización de la pantalla producida por el programa de la figura 1.5.

En los siguientes capítulos del libro, exploraremos las posibilidades de los gráficos de alta y baja resolución, y descubriremos las técnicas de dibujo de formas, coloreado, creación de nuevos símbolos y producción de diagramas o gráficos. También veremos algunos de los principios de la animación y de la creación de perspectiva y diagramas pseudo-tridimensionales.

Capítulo 2

Gráficos de baja resolución

Comencemos nuestra exploración de los gráficos creados con el Spectrum, estudiando los gráficos de *BAJA RESOLUCION*, que utilizan la pantalla de símbolos de texto.

Utilización de los espacios de símbolos de texto

En los primeros días de los ordenadores, toda salida de ordenador se imprimía en una impresora utilizando símbolos de texto, y no existían facilidades para la obtención de gráficos en pantalla. Los programadores soslayaban esta limitación utilizando símbolos de texto como elementos individuales de dibujo, en creación de imágenes y gráficos. Los símbolos como M ó W producen gamas gris oscuro, mientras que los puntos son elementos muy claros. Eligiendo cuidadosamente el tipo y la posición de los símbolos de texto, se puede crear un dibujo sobre papel, que tendrá un aspecto bastante bueno si se mira desde suficiente distancia para no ver individualmente los símbolos de texto que lo componen. Con esta técnica se pueden realizar dibujos de animales, o retratos de personalidades, de calidad aceptable. Esta técnica no funciona muy bien en la pantalla del Spectrum, pero puede utilizarse conectando al ordenador una impresora de 80 columnas mediante un interface, y utilizando unas 100 líneas de texto para la realización del dibujo. Esto nos daría unos 8.000 elementos individuales de diseño para un área de 8 x 10 pulgadas de papel. Si se observa desde unos ocho pies de distancia, el dibujo resultante puede ser muy expresivo. Una técnica muy similar a ésta es la que se utiliza para imprimir dibujos con computadora sobre camisetas de sport.

Para producir líneas horizontales, puede utilizarse el signo "—" o el signo de subrayar. Las líneas verticales se pueden obtener utilizando la letra I mayúscula, repitiéndo las I unas sobre las otras en la misma columna de impresora.

Símbolos gráficos de mosaico

Estudiemos ahora con más detenimiento el conjunto de símbolos gráficos que puede producir el Spectrum, ejecutando el programa de la figura 2.1.(a). Con él, produciremos todos los símbolos con código de carácter, desde 32 hasta 164.

Los códigos desde ∅ hasta 32 se utilizan para funciones de control, y normalmente no producen símbolos en pantalla. Los códigos por encima de 164 se utilizan por el intérprete de BASIC del Spectrum, cuando traduce las instrucciones de un programa, para listarlo. Estas instrucciones están almacenadas en forma de palabras impresas. Para ahorrar espacio en la memoria, los comandos BASIC se almacenan en la memoria como palabras de un solo dato, llamadas *TOKENS*. Por lo tanto, los códigos de caracteres por encima del 164 imprimen palabras de comando tales como, por ejemplo, PRINT, GOSUB y otras.

```
100 REM Imprime el juego de caracteres
105 REM con codigos del 32 al 164
120 CLS
130 PRINT
140 FOR n=32 TO 164
150 PRINT CHR$ n;" ";
160 NEXT n
```

Fig. 2.1.(a) Programa que imprime el conjunto de caracteres standard del Spectrum.

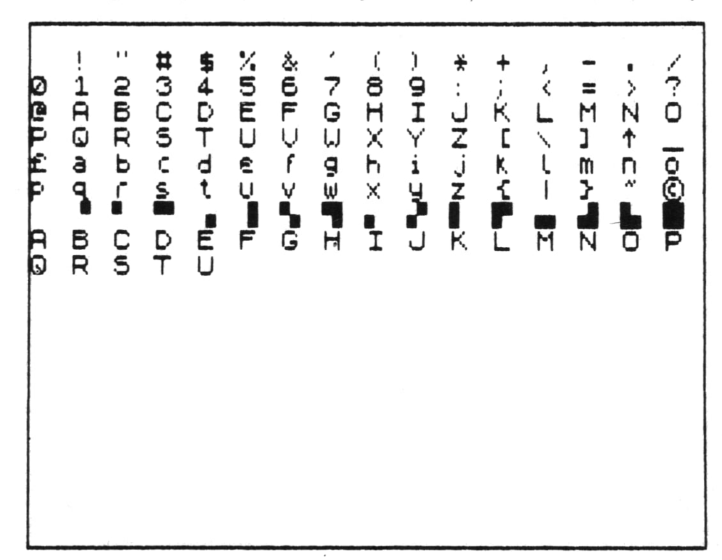


Fig. 2.1.(b) Imagen producida por el programa de la figura 2.1.(a).

La primera acción del programa de la figura 2.1.(a) es limpiar la pantalla, utilizando para ello la instrucción CLS. Esta instrucción deja blanca la zona de trabajo de la pantalla. A continuación se imprimen los símbolos, utilizando para ello un sencillo bucle (loop) y una instrucción PRINT CHR$n. La instrucción CHR$n imprime en la pantalla los símbolos cuyo código de carácter sea n. Fíjese en el punto y coma del final de la línea. Si no se incluyera el Spectrum, imprimiría todos los símbolos, pero los visualizaría uno por línea, y al alcanzar los 22 símbolos sería necesario un desplazamiento de pantalla, pulsando la tecla Y para ver los siguientes 22 símbolos. El punto y coma permite que los caracteres de las instrucciones PRINT sucesivas se impriman uno detrás del otro en la misma línea.

Además del conjunto de los símbolos de texto, usted se habrá fijado que tiene un conjunto de símbolos formados por pequeños bloques. Este conjunto es el de los símbolos gráficos de mosaico. Observará que en este bloque de símbolos gráficos el espacio del símbolo está dividido en cuatro segmentos, y cada segmento puede ser negro o blanco, formando un total de dieciséis patrones diferentes. Cada patrón tiene un número de código de carácter, en este caso, desde el 128 al 143. Para producir uno cualquiera de estos dieciséis patrones de bloque, elegiremos sencillamente el código de carácter apropiado, y utilizaremos la expresión CHR$ en una instrucción PRINT.

La figura 2.2. muestra el conjunto completo de gráficos de mosaico, con sus respectivos códigos de carácter.

▢	128	▖	136
▝	129	▞	137
▘	130	▌	138
▀	131	▛	139
▗	132	▄	140
▐	133	▟	141
▚	134	▙	142
▜	135	█	143

Fig. 2.2. Gráficos de mosaico y sus códigos de carácter asociados.

Otra forma alternativa para imprimir los símbolos gráficos es utilizar el teclado en modo gráfico. Podemos entrar a este modo pulsando las teclas CAPS SHIFT y 9 a la vez. Si está en modo gráfico, el cursor intermitente de la pantalla cambiará de L a G. Para volver al modo normal, vuelva a presionar a la vez las teclas CAPS SHIFT y 9.

Una vez en modo gráfico, si presiona las teclas de la primera fila, que llevan los dibujos de los símbolos gráficos en blanco, se imprimirán en la pantalla todos los símbolos gráficos como si fueran símbolos normales de texto. Sólo disponemos de ocho teclas de símbolos gráficos, por tanto, para obtener los ocho siguientes patrones, presione CAPS SHIFT y la tecla del símbolo correspondiente, exactamente igual que si tecleara una mayúscula. Para escribir símbolos gráficos en un programa, colóquelos después de las comillas, exactamente igual que si fueran símbolos de texto normales.

Visualización con colores

El sistema de visualización del Spectrum, en blanco y negro, es muy útil para visualizar texto, ya que es similar al texto impreso sobre papel. Sin embargo, el Spectrum es capaz de darnos visualizaciones llenas de color, y éstas son las que nos resultarán especialmente atractivas para la creación de gráficos.

Para cambiar el color de los puntos que forman un símbolo en la pantalla, utilizaremos el comando INK, que se obtiene presionando CAPS SHIFT y SYMBOL SHIFT al mismo tiempo, para pasar así al modo extendido, y pulsar después las teclas X y CAPS SHIFT a la vez. La instrucción INK va seguida de un número del Ø al 9, que selecciona uno de los nueve colores posibles.

Cualquier nuevo símbolo que se escriba ahora en la pantalla, aparecerá en el nuevo color seleccionado, pero todos los demás signos que estaban no se alteran.

La instrucción INK seguida por un determinado número (del Ø al 9), puede producir los siguientes colores:

0	Negro	5	Azul Cyan
1	Azul	6	Amarillo
2	Rojo	7	Blanco
3	Magenta	8	Transparente
4	Verde	9	Contraste

El color cyan es un azul verdoso pálido, producido efectivamente al mezclar en la pantalla el azul y el verde. El Magenta (rojo + azul) y el amarillo (verde + rojo) se producen mezclando rojo con azul o verde.

Los colores con número 8 y 9 son diferentes de los demás, ya que no especifican directamente un color de tinta (INK). Recordará que vimos, en el capítulo 1, que el Spectrum coloreaba cada uno de los espacios de los símbolos y almacenaba esta información sobre el color en una zona de la memoria diferente de la de los patrones de puntos. Cuando conecta el Spectrum, se asigna tinta negra (INK Ø), como color de tinta, a todas las posiciones de caracteres.

Si introduce INK 8, el símbolo se imprimirá en cualquier color que tuviera asignado ese espacio de caracteres particular de la pantalla. Esto puede resultar útil cuando se tienen diferentes zonas de colores en la pantalla. Una vez realizada la visualización inicial, la nueva información puede imprimirse en el color INK 8, y el color dependerá del lugar de la pantalla en el que se imprima. Así, se pueden actualizar los elementos inmediatamente, sin tener que decidir el color, y luego se pueden cambiar al color deseado.

Si imprimimos un color claro, como, por ejemplo, el amarillo sobre un fondo blanco, la lectura se hará difícil, porque los dos colores son similares. Si se utiliza el color 9, el ordenador elegirá automáticamente un color para el texto, que será blanco o negro, de acuerdo con el color de fondo (claro u oscuro). De este modo, los símbolos de texto siempre contrastarán con el fondo y se leerán con facilidad.

Para ver el efecto del color en la visualización, hagamos funcionar el programa siguiente, que produce unos patrones muy sencillos en una gama de colores. Utilizaremos los gráficos de mosaico y algunos símbolos de texto. Este programa se lista en la figura 2.3., y los patrones que ofrece pueden verse en la figura 2.4.

```
100 REM  Patrones de papel pintado.
110 DIM a(7)
120 DIM c(7)
130 FOR s=1 TO 30
140 CLS
150 REM Prepara el patron.
160 FOR p=1 TO 7
170 LET a(p)=122+INT (RND*22)
180 LET c(p)=INT (7*RND)
190 NEXT p
200 REM Imprime el patron.
210 LET k=3+INT (5*RND)
220 FOR n=1 TO 672 STEP k
```

```
230 FOR j=1 TO k
240 PRINT INK c(j);CHR$ a(j);
250 NEXT j
260 NEXT n
270 PAUSE 200
280 NEXT s
```

Fig. 2.3. Programa que produce un patrón de papel pintado.

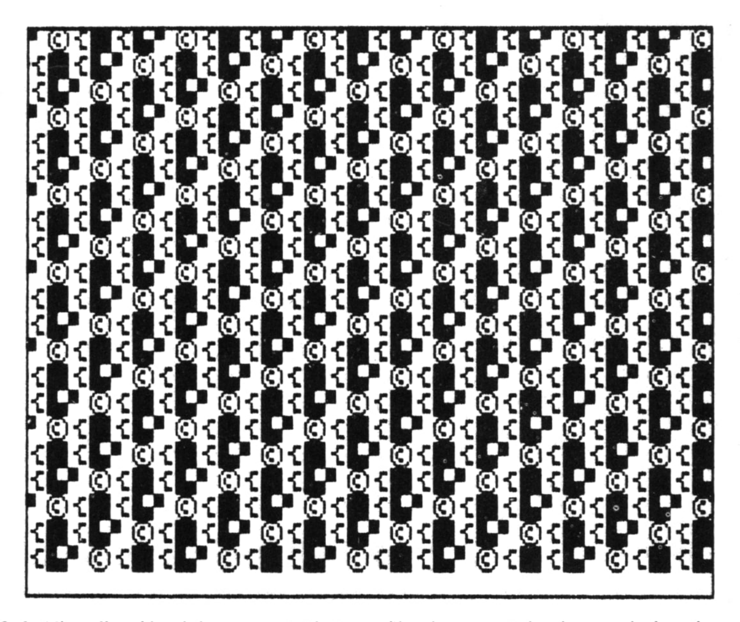


Fig. 2.4. Visualización del programa de creación de un patrón de papel pintado.

El esquema básico de este programa es el siguiente: el ordenador genera aleatoriamente un conjunto de siete caracteres y toma un grupo de ellos, de tres a siete, para imprimirlos repetidamente en colores diferentes hasta llenar la pantalla. Cada patrón permanece en la pantalla unos segundos (para ello se ha utilizado la instrucción PAUSE), y luego desaparece para dibujarse un nuevo patrón. El programa genera un conjunto de 20 patrones, pero podría seguir produciéndolos casi indefinidamente, si se incrementara el tamaño del bucle que utiliza las variables.

Cambio del color de fondo

El Spectrum arranca con color de fondo blanco, texto negro y un borde blanco rodeando la zona de visualización. Si utilizamos la

instrucción CLS de limpieza de pantalla, se borrarán todos los símbolos existentes y se restaurará el fondo blanco.

Para cambiar este blanco, podemos producir una gama de colores de fondo para la pantalla, utilizando la instrucción PAPER seguida de otra CLS. La instrucción PAPER se obtiene del mismo modo que la instrucción INK, excepto en el modo extendido, que hay que utilizar la tecla C. Los números de color colocados después de PAPER tienen exactamente el mismo significado que tenían para los colores de tinta (INK).

En el momento de fijar el color de fondo con PAPER y CLS, la pantalla queda vacía, por tanto es importante que realice esta operación antes de comenzar a imprimir algún texto o dibujo.

En lugar de utilizar PAPER para cambiar el color de toda la pantalla, podemos usarlo también para fijar el color de fondo de símbolos individuales. De este modo, si utilizamos la instrucción PAPER 2, todos los símbolos que se impriman a partir de ese momento tendrán un color de fondo rojo. Este color de fondo sólo aparecerá en los espacios en los que se escriban símbolos nuevos, y ninguno de los colores de fondo de los símbolos anteriormente escritos se verá afectado.

Comandos de color en la instrucción PRINT

Algunas veces desearemos escribir sólo dos o tres símbolos en un nuevo color, para volver al color normal de tinta (INK). Evidentemente, esto puede realizarse insertando comandos INK, para cambiar el color, antes y después de la impresión de los símbolos. Un método mejor consiste en incluir el comando INK en la instrucción PRINT del modo siguiente:

```
200 PRINT INK2; "texto rojo"
```

de esta forma, el color INK se alterará temporalmente, cambiando a rojo durante el mensaje.

El texto se imprimirá, efectivamente, en rojo, pero una vez escrito, se vuelve el color de tinta (INK) anterior (cualquiera que fuera el color fijado por el programa). Esta misma técnica también puede utilizarse con PAPER y cambiar el color de fondo de unos pocos símbolos. Observe que cuando utilice INK o PAPER del modo anteriormente descrito, debe utilizar un punto y coma para separar el comando del resto de la instrucción PRINT.

La figura 2.5. muestra una versión modificada del programa del papel pintado, utilizando cambios de color con INK y PAPER. El resultado es aún más colorista.

```
100 REM Patronesde papel pintado especiales
110 REM con cambio de color en el fondo-PAPER-y
en el texto-INK-.
120 DIM a(7)
130 DIM c(7)
140 DIM p(7)
150 FOR s=1 TO 30
160 CLS
170 FOR i=1 TO 7
180 LET a(i)=122+INT (RND*22)
190 LET c(i)=INT (8*RND)
200 LET p(i)=INT (RND*8)
210 IF p(i)=c(i) THEN GO TO 185
220 NEXT i
230 REM Impresion del patron
240 LET k=3+INT (5*RND)
250 FOR n=1 TO 672 STEP k
260 FOR j=1 TO k
270 PRINT INK c(j); PAPER p(j);CHR$ a(j);
280 NEXT j
290 NEXT n
300 PAUSE 200
310 NEXT s
```

Fig. 2.5. Programa para obtener un mejor diseño de papel pintado.

Colocación de símbolos utilizando PRINT AT

Hasta el momento hemos imprimido filas de símbolos en la pantalla en el lugar en el que se encontraba el cursor. Para gráficos más serios, sería muy útil si pudiéramos colocar directamente un símbolo en cualquier espacio disponible de la pantalla.

Con el modo de texto, se pueden colocar símbolos en cualquier espacio de símbolo disponible de la pantalla, utilizando la instrucción PRINT AT. Esta instrucción tiene la forma siguiente:

```
1ØØ PRINT AT r,c;"texto"
```

donde r y c son las coordenadas que especifican el punto de la pantalla donde queremos visualizar el símbolo. El valor de r puede variar entre Ø y 21, y especifica la fila en la que va a imprimirse el texto, comenzando por la fila Ø en la parte superior de la pantalla, y bajando hasta la fila 21 al final del área de visualización. Observe

que PRINT AT no imprime texto en las dos últimas filas (22 y 23), que son las reservadas para los mensajes del ordenador. La variable c indica el lugar en el que se escribirá el texto, comenzando a contar en la pantalla desde el borde izquierdo hasta la posición 31, en el borde derecho de la pantalla.

Con 22 filas y 32 columnas por fila, hay en la pantalla un total de 704 espacios de símbolo. Si empezamos en el ángulo superior izquierdo, sus coordenadas r,c serán Ø,Ø. Si nos movemos por cada fila, c va aumentando desde Ø a 31, y si vamos bajando hacia el final, r aumenta desde Ø hasta 21, como puede verse en la figura 2.6.

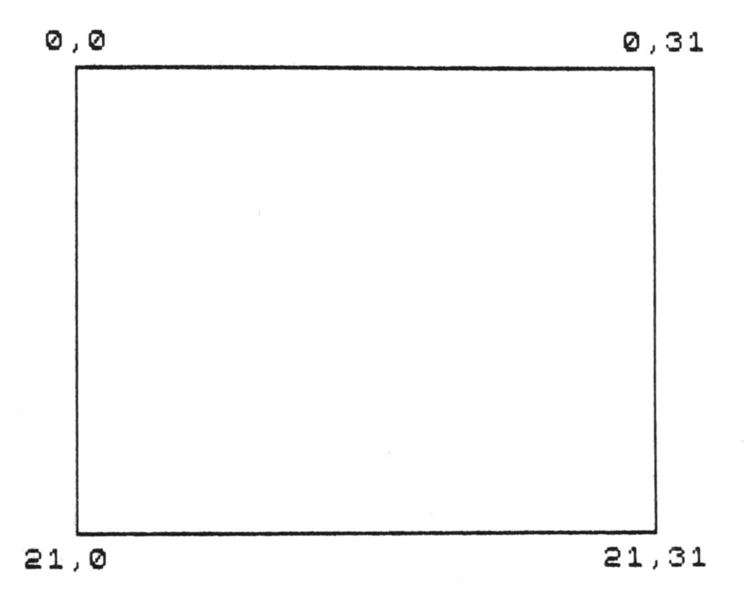


Fig. 2.6. Esquema de la pantalla a utilizar cuando se imprime con PRINT AT.

El programa que se lista en el figura 2.7. selecciona códigos de carácter aleatoriamente, y luego imprime el símbolo de carácter correspondiente en la pantalla en una posición también aleatoria. En este caso, los números c y r se generan multiplicando la función RND por 32 y 22, respectivamente. La función RND produce un número entre Ø y 1, aunque nunca produce el 1. Los valores de c y r deben ser enteros, y, por tanto, aplicaremos la operación INT a los resultados. El efecto de INT es eliminar la parte fraccionaria del número, dejando sólo la parte entera. Como RND no alcanza nunca 1, el resultado del cálculo de c es un número que va desde Ø a 31, que es justo lo que necesitamos para localizar la posición de la columna

número c. El valor de r se redondea del mismo modo para dar un número en la gama del ∅ al 21.

```
100 REM Impresion de  simbolos
utilizando PRINT AT
110 REM en posiciones elegidas de un modo
aleatorio.
120 FOR n=1 TO 100
125 REM Establece la posicion de fila y columna.
130 LET r=INT (RND*22)
135 REM Marca el color del simbolo.
140 LET c=INT (RND*32)
150 INK INT (RND*7)
155 REM Selecciona el color del simbolo.
160 LET s=96+INT (RND*48)
165 REM Imprime el simbolo en la pantalla.
170 PRINT AT r,c;CHR$ s;
180 NEXT n
```

Fig. 2.7. Programa para dibujar símbolos en posiciones aleatorias utilizando PRINT AT.

Puntos individuales del mosaico

Como el sistema de los gráficos de mosaico divide los espacios de caracteres en dos filas y dos columnas, tendremos, pues, una pantalla gráfica de 64 puntos de anchura y 44 de altura. Para que sea verdaderamente efectiva, deberíamos ser capaces de activar o desactivar los estados de los bloques individuales del interior del espacio de caracteres.

Si examinamos el código de los símbolos de mosaico, veremos que cada bloque del espacio tiene un valor numérico que es el que se muestra en la figura 2.8. Para saber el código de carácter de un patrón particular de bloques, sume simplemente los valores de los bloques activados (color de tinta INK) y añada el resultado a 128.

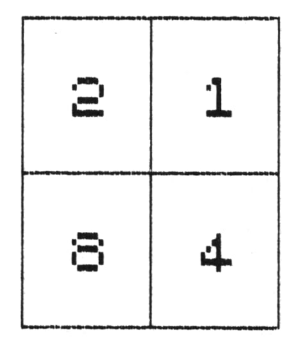


Fig. 2.8. Valores numéricos de los bloques de símbolos gráficos de mosaico.

En nuestra pantalla de 64 x 44 puntos, podemos definir la posición de un punto utilizando la coordenada x para medir su posición horizontal en la pantalla, comenzando desde el borde izquierdo. Y la coordenada y, para obtener su posición en sentido vertical, comenzando desde la parte superior. De este modo, x toma valores desde Ø a 63, é y los toma de Ø a 43.

Para encontrar la posición (fila y columna) del símbolo de mosaico, dividiremos sencillamente x é y por 2, y tomaremos la parte entera. Como ejemplo, si elegimos x = 33 e y = 25, el valor r de la fila será

r = INT(y/2) = INT (12.5) = 12

y la columna c será

c = INT (x/2) = INT (16.5) = 16

Ahora nos queda elegir el símbolo correcto para que el punto esté en el lugar adecuado. Comenzando con la posición x, si c = x/2, el bloque debe estar a la izquierda del espacio del carácter, y esto le da al patrón un valor de 2. Si c <> x/2, el bloque estará en la derecha y el valor del patrón será 1. En la dirección de y, si r = y/2, el bloque está en la parte superior del espacio del carácter, y el valor del patrón es aquel que antes calculamos a partir de la posición x. En caso contrario, el bloque se encontrará en la parte inferior del espacio, y el valor del patrón debe multiplicarse por cuatro para dar el patrón correcto.

Podemos utilizar la siguiente rutina corta para convertir los valores x é y en códigos y posiciones de carácter:

```
500 LET c=INT (x/2)
510 LET r=INT (y/2)
520 LET p=1
530 IF c=x/2 THEN LET p=p*2
540 IF r<>y/2 THEN LET p=p*4
550 PRINT AT r,c;CHR$ (128+p);
560 RETURN
```

Esta rutina funcionará bien con la pantalla vacía, pero tendrá problemas en el caso de que se desee colocar un punto en la posición de un carácter donde ya había otro punto. El nuevo símbolo de mosaico que imprimimos borrará cualquier símbolo que estuviera antes en esa posición. Por tanto, necesitamos ser capaces de saber

qué puntos del interior del espacio del símbolo están activados, y añadir el nuevo punto, para producir un nuevo símbolo en la impresión, si es necesario. Como el Spectrum almacena patrones de puntos, y no códigos de caracteres, en la memoria de visualización, no podemos utilizar simplemente PEEK para descubrir qué símbolo se está escribiendo, como haríamos con ordenadores de otros tipos. De hecho, el Spectrum tiene una instrucción llamada SCREEN$ (pantalla) que suministra el código de carácter para el símbolo escrito en cualquier posición de la pantalla. El único problema es que no funciona con los códigos de los gráficos de mosaico.

No perdamos la esperanza, sin embargo, ya que podemos adoptar el mismo tipo de esquema que utilizan otros ordenadores, y crear una matriz p (c,r) que almacenará todos los códigos de símbolo que se impriman en la pantalla. Una vez hecho esto, llamando la fila y columna adecuada en la matriz, podemos encontrar el código de cualquier posición de carácter.

En la matriz almacenamos los valores que tienen en ese momento los bloques de los símbolos gráficos, que será el código de carácter menos 128. Comprobaremos el código de patrón para el símbolo que se encuentre en la posición que hemos seleccionado, y veremos si el punto está activado o no. Si está activado, añadiremos 128 al código de patrón p(c,r), y tendremos el código de carácter para la impresión.

Si el punto no está activado, se añade el código de ese punto a p(c,r), y se almacena este nuevo valor. Este código es el que se utiliza en la producción del código de carácter para la impresión.

En este momento, ya podemos producir una subrutina que dibujará un punto en una posición x,y de la pantalla de gráficos de mosaico. Esta subrutina se ha utilizado en el programa de la figura 2.9. para dibujar un patrón de puntos sobre la pantalla, de forma aleatoria.

```
100 REM Puntos aleatorios con graficos de
mosaico.
110 DIM p(32,22)
120 PRINT AT 0,0;"Creacion de la matriz";
130 FOR y=1 TO 22: FOR x=1 TO 32
140 LET p(x,y)=0
150 NEXT x: NEXT y: CLS
160 REM Dibuja los puntos
170 FOR n=1 TO 500
180 LET x=INT (RND*64)
190 LET y=INT (RND*44)
200 INK INT (RND*8)
210 GO SUB 500
```

```
220 NEXT n
230 STOP
490 REM Subrutina de dibujo de puntos.
500 LET r=INT (y/2)
510 LET c=INT (x/2)
520 LET p1=1
530 IF c=x/2 THEN LET p1=p1*2
540 IF r<>y/2 THEN LET p1=p1*4
550 LET p2=p(c+1,r+1)
560 IF p2<8 AND p1=8 THEN GO TO 640
570 IF p2>=8 THEN LET p2=p2-8
580 IF p2<4 AND p1=4 THEN GO TO 640
590 IF p2>=4 THEN LET p2=p2-4
600 IF p2<2 AND p1=2 THEN GO TO 640
610 IF p2>=2 THEN LET p2=p2-2
620 IF p2<1 AND p1=1 THEN GO TO 640
630 GO TO 650
640 LET p(c+1,r+1)=p(c+1,r+1)+p1
650 PRINT AT r,c;CHR$ (128+p(c+1,r+1));
660 RETURN
```

Fig. 2.9. Programa que presenta puntos aleatoriamente. Para ello utiliza símbolos gráficos de mosaico.

Dibujo de líneas

Para dibujar una línea sobre una pantalla de baja resolución, podemos trabajar con las posiciones de los puntos que deben encenderse sobre una hoja de papel que represente la rejilla de la pantalla. Cada cuadrado que atraviese la línea será uno de los que debe encenderse. Podemos leer sus coordenadas, que serán las utilizadas en la subrutina de dibujo de puntos, para coiorear los puntos adecuados y dibujar una línea sobre la pantalla. Este proceso es muy tedioso, y resulta más sencillo dejar que sea el ordenador el que trabaje de forma automática. Todo lo que necesitamos es decirle al ordenador cuáles son las coordenadas x é y de los dos puntos finales de la línea deseada. El dibujo de líneas horizontales y verticales es relativamente simple.

Supongamos que deseamos dibujar una línea horizontal entre dos puntos x1 y x2. En este caso, como la línea es horizontal, la coordenada y será la misma en los dos extremos de la línea. Como cada punto representa un paso en el valor de x, el número de puntos de dibujar será la diferencia entre x1 y x2.

La figura 2.10. muestra un programa que dibuja una línea horizontal. El cambio del valor de x entre dos puntos sucesivos a lo largo de la línea, debe ser 1, puesto que los pasos de las x van de uno en uno (xs=1). Si suponemos que x1 y x2 pueden estar en cualquier

lugar de la pantalla, el resultado del cálculo x2—x1 puede ser negativo, y en este caso será xs=—1. El número de puntos de la línea es n =ABS(x2—x1). Como estamos utilizando n en un bucle, la función ABS se utiliza para el caso de que x2—x1 sea negativo, ya que n seguirá siendo positiva.

La línea se dibuja con bucle muy sencillo, y por cada paso por el bucle (desde uno hasta ns) se escribe un punto en la pantalla. La coordenada x de cada punto se calcula añadiendo s*xs al valor de x1. Se comienza en Ø, de forma que el primer punto se dibuja en x1.

El dibujo de una línea vertical exige un procesamiento muy similar, pero esta vez los valores de y son los que cambian, según nos vamos moviendo por la línea, y las x permanecen constantes. El programa para dibujar la línea vertical se muestra en la figura 2.11. En efecto, sólo hemos necesitado transponer en el programa los términos x é y.

Un punto a notar es que, si dibujamos dos líneas que se cruzan con distintos colores de tinta, en los puntos de cruce la línea original puede cambiar de color. Esto se debe a la restricción de que sólo puede haber un color de tinta para cada posición de carácter en la pantalla de texto. Recuerde que, aunque pintamos puntos individuales, de hecho sólo hay bloques en el interior de los símbolos mosaico, y un símbolo tiene todos los bloques del mismo color. Si se escribe un nuevo punto en el interior de un espacio donde estaba ya activado otro bloque con un color de tinta (INK), todos los bloques cambian de color, pasando al nuevo color de tinta (INK).

```
100 REM Lineas horizontales con graficos de
mosaico.
110 DIM p(32,22)
120 PRINT AT 0,0;"Creacion de la matriz";
130 FOR y=1 TO 22: FOR x=1 TO 32
140 LET p(x,y)=0
150 NEXT x: NEXT y: CLS
160 REM Dibuja las lineas.
170 FOR n=1 TO 50
175 REM Establece los puntos de comienzo y final.
180 LET x1=INT (RND*64)
190 LET x2=INT (RND*64)
200 LET y1=INT (RND*44)
205 REM Establece la direccion de trazado.
210 LET xs=SGN (x2-x1)
215 REM Establece el numero de pasos.
220 LET ns=ABS (x2-x1)
230 FOR s=1 TO ns
240 LET x=x1+s*xs
250 LET y=y1
260 GO SUB 500
270 NEXT s
```

```
280 INK INT (RND*7)
290 NEXT n
300 STOP
490 REM Subrutina de dibujo de puntos.
500 LET r=INT (y/2)
510 LET c=INT (x/2)
520 LET p1=1
530 IF c=x/2 THEN LET p1=p1*2
540 IF r<>y/2 THEN LET p1=p1*4
550 LET p2=p(c+1,r+1)
560 IF p2<8 AND p1=8 THEN GO TO 640
570 IF p2>=8 THEN LET p2=p2-8
580 IF p2<4 AND p1=4 THEN GO TO 640
590 IF p2>=4 THEN LET p2=p2-4
600 IF p2<2 AND p1=2 THEN GO TO 640
610 IF p2>=2 THEN LET p2=p2-2
620 IF p2<1 AND p1=1 THEN GO TO 640
630 GO TO 650
640 LET p(c+1,r+1)=p(c+1,r+1)+p1
650 PRINT AT r,c;CHR$ (128+p(c+1,r+1));
660 RETURN
```

Fig. 2.10. Programa para dibujar líneas horizontales con símbolos mosaico.

```
100 REM Lineas verticales con graficos de mosaico.
110 DIM p(32,22)
120 PRINT AT 0,0;"Creacion de la matriz";
130 FOR y=1 TO 22: FOR x=1 TO 32
140 LET p(x,y)=0
150 NEXT x: NEXT y: CLS
160 REM Dibuja las lineas.
170 FOR n=1 TO 50
175 REM Establece los puntos de comienzo y final.
180 LET x1=INT (RND*64)
190 LET y1=INT (RND*44)
200 LET y2=INT (RND*44)
205 REM Establece la direccion de trazado.
210 LET ys=SGN (y2-y1)
215 REM Establece el numero de pasos.
220 LET ns=ABS (y2-y1)
230 FOR s=1 TO ns
240 LET x=x1
250 LET y=y1+s*ys
260 GO SUB 500
270 NEXT s
280 INK INT (RND*7)
290 NEXT n
300 STOP
490 REM Subrutina de dibujo de puntos.
500 LET r=INT (y/2)
510 LET c=INT (x/2)
520 LET p1=1
530 IF c=x/2 THEN LET p1=p1*2
540 IF r<>y/2 THEN LET p1=p1*4
550 LET p2=p(c+1,r+1)
560 IF p2<8 AND p1=8 THEN GO TO 640
```

```
570 IF p2>=8 THEN LET p2=p2-8
580 IF p2<4 AND p1=4 THEN GO TO 640
590 IF p2>=4 THEN LET p2=p2-4
600 IF p2<2 AND p1=2 THEN GO TO 640
610 IF p2>=2 THEN LET p2=p2-2
620 IF p2<1 AND p1=1 THEN GO TO 640
630 GO TO 650
640 LET p(c+1,r+1)=p(c+1,r+1)+p1
650 PRINT AT r,c;CHR$ (128+p(c+1,r+1));
660 RETURN
```

Fig. 2.11. Programa para dibujar líneas verticales utilizando signos gráficos de mosaico.

Programa sencillo de bosquejo

La producción de dibujos con una pantalla de baja resolución puede ser un trabajo muy laborioso, ya que exige la realización del dibujo sobre un papel adecuado, con 32 espacios de ancho y 22 de alto. Cada uno de estos cuadrados se subdivide en cuatro, y los bloques individuales de dentro de los espacios de los símbolos se rellenan para obtener el dibujo deseado. Los símbolos de cada fila se convierten en una cadena de símbolos de texto, y finalmente se imprimen en la pantalla para producir el dibujo.

Otra técnica alternativa para la producción de dibujos en pantalla, es la utilización de un programa sencillo de bosquejo que funciona de forma similar a la máquina de grabar. En este tipo de máquina, una pluma se mueve en las dos direcciones, vertical y horizontal, sobre una hoja de papel, mediante dos tiradores o palancas. La pluma puede, a su vez, estar apoyada y dibujar una línea, o bien separada en movimiento por el papel.

Es muy sencillo realizar un programa en el que se controle el movimiento de la pluma por la pantalla, utilizando las teclas de flechas de la parte superior del Spectrum. El cambio de estado de la pluma (hacia arriba o hacia abajo) puede controlarse con las teclas U y D.

Una de las primeras cosas que necesitamos es saber detectar qué tecla del teclado se ha pulsado, y, de este modo, tomar la acción adecuada. Esto podemos conseguirlo utilizando la instrucción INKEY$. Esta instrucción devolverá el número que corresponde al código de carácter de la tecla que se ha presionado. Sin embargo, esta instrucción no espera a que usted teclee una tecla, simplemente comprueba si se ha tecleado alguna en el momento de ejecutarse. Para vigilar el teclado, necesitaremos una acción de bucle constante, utilizando la instrucción siguiente:

210 LET a$=INKEY$:IFa$=“ ”THEN GO TO 210

En este caso, la cadena a$ toma el valor del código producido por INKEY$. Si no apretamos ninguna tecla, a$ será una cadena vacía (“ ”), y la instrucción vuelve sobre sí misma y se repite continuamente. Si se ha apretado una tecla, a$ no estará vacía, y la prueba falla, de forma que el programa salta a la instrucción siguiente.

Si se ha detectado que se presionó una tecla, habrá que examinar los valores de a$ para detectar cual fue la tecla que se pulsó. Podemos empezar a comprobar las teclas de flechas. Estos códigos de flecha (previstos para el modo de operación con la tecla de mayúsculas —SHIFT—) sólo se obtienen cuando se pulsan estas teclas a la vez que la de CAPS SHIFT. En el Spectrum, las teclas de flecha son también los números 5,6,7 y 8. En el modo de teclado normal, por tanto, estas teclas producen los códigos de los números 5,6,7 y 8, del modo siguiente:

Flecha izquierda	5
Flecha derecha	8
Flecha inferior	6
Flecha superior	7

Si se detecta una flecha derecha, el valor de x se incrementa en 1, y, si se detecta la flecha izquierda, se disminuye en 1. El valor y se incrementa si se detecta la flecha inferior, y se decrementa si la flecha es superior. Se han hecho pruebas para detectar los valores de x inferiores a 0 o superiores a 63, ya que estos valores causan errores en la instrucción PRINT AT. Si x toma un valor inferior a 0, se le fijará un valor 0, y si es superior a 63, se fijará en 63. Esto da un efecto de corte, como si la línea se parase a un lado de la pantalla. Para el límite vertical de los valores de y, se hacen comprobaciones semejantes. Si se detecta la tecla D, fijaremos la variable w en 1, para indicar que la pluma está hacia abajo. Si se detecta U, w se fija en 0, y nos indica que la pluma está hacia arriba.

```
100 REM Programa de bosquejo para graficos de mosaico.
120 PRINT AT 0,0;"Creacion de la matriz de pantalla";
130 DIM p(32,22)
140 FOR y=1 TO 22: FOR x=1 TO 32
150 LET p(x,y)=0
160 NEXT x: NEXT y: CLS
170 LET x1=32: LET x2=32
180 LET y1=22: LET y2=22
190 LET w=1: LET x=x1: LET y=y1: GO SUB 700
```

```
 195 REM Bucle de bosquejo.
 200 PRINT AT 0,0;"x=";x1;" y=";y1;"      ";
 210 LET a$=INKEY$: IF a$="" THEN GO TO 210
 220 IF a$="5" THEN LET x1=x1-1: GO TO 300
 230 IF a$="8" THEN LET x1=x1+1: GO TO 300
 240 IF a$="6" THEN LET y1=y1+1: GO TO 300
 250 IF a$="7" THEN LET y1=y1-1: GO TO 300
 260 IF a$="d" THEN LET w=1: GO TO 300
 270 IF a$="u" THEN LET w=0: GO TO 300
 280 IF a$="q" THEN STOP
 290 GO TO 200
 295 REM Compara x e y con los limites.
 300 IF x1<0 THEN LET x1=0
 310 IF x1>63 THEN LET x1=63
 320 IF y1<2 THEN LET y1=2
 330 IF y1>43 THEN LET y1=43
 340 LET lp=pc
 350 LET x=x1: LET y=y1: GO SUB 700
 360 LET x=x2: LET y=y2
 370 IF w=0 AND lp=1 THEN GO SUB 900
 380 LET x2=x1: LET y2=y1
 390 GO TO 200
 690 REM Activa la subrutina de puntos.
 700 LET r=INT (INT (y)/2)
 710 LET c=INT (INT (x)/2)
 720 LET p1=1: LET pc=0
 730 IF c=INT (x)/2 THEN LET p1=p1*2
 740 IF r<>INT (y)/2 THEN LET p1=p1*4
 750 LET p2=p(c+1,r+1)
 760 IF p2<8 AND p1=8 THEN GO TO 840
 770 IF p2>=8 THEN LET p2=p2-8
 780 IF p2<4 AND p1=4 THEN GO TO 840
 790 IF p2>=4 THEN LET p2=p2-4
 800 IF p2<2 AND p1=2 THEN GO TO 840
 810 IF p2>=2 THEN LET p2=p2-2
 820 IF p2<1 AND p1=1 THEN GO TO 840
 830 GO TO 850
 840 LET p(c+1,r+1)=p(c+1,r+1)+p1
 845 LET pc=1
 850 PRINT AT r,c;CHR$ (128+p(c+1,r+1));
 860 RETURN
 895 REM Borra la subrutina de puntos.
 900 LET r=INT (INT (y)/2)
 910 LET c=INT (INT (x)/2)
 920 LET p1=1
 930 IF c=INT (x)/2 THEN LET p1=p1*2
 940 IF r<>INT (y)/2 THEN LET p1=p1*4
 950 LET p2=p(c+1,r+1)
 960 IF p2>=8 AND p1=8 THEN GO TO 1040
 970 IF p2>=8 THEN LET p2=p2-8
 980 IF p2>=4 AND p1=4 THEN GO TO 1040
 990 IF p2>=4 THEN LET p2=p2-4
1000 IF p2>=2 AND p1=2 THEN GO TO 1040
1010 IF p2>=2 THEN LET p2=p2-2
1020 IF p2>=1 AND p1=1 THEN GO TO 1040
1030 GO TO 1050
1040 LET p(c+1,r+1)=p(c+1,r+1)-p1
1050 PRINT AT r,c;CHR$ (128+p(c+1,r+1));
1060 RETURN
```

Fig. 2.12. Un programa de bosquejo sencillo, utilizando gráficos de mosaico.

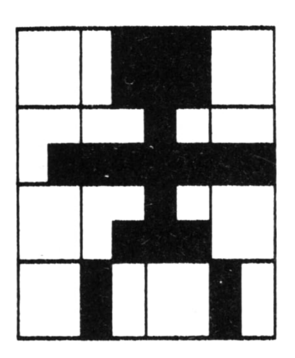
Si la pluma está hacia abajo, la subrutina de la línea 500 dibujará un punto en la nueva posición. Si la pluma está hacia arriba, se salta la subrutina o se borra el punto. Las posiciones x é y de la pluma se imprimen en la parte superior de la pantalla. El listado del programa se muestra en la figura 2.12.

Producción de dibujos

Hasta el momento hemos creado patrones, escrito puntos individuales, dibujado líneas y trazado sobre la pantalla, pero, con frecuencia, lo que usted desea es realizar un dibujo. La mejor manera de hacerlo es colocar el dibujo sobre una rejilla, y trabajar con los símbolos que se deben imprimir para producir el resultado apetecido.

Supongamos que queremos producir un dibujo de un monigote. Comenzaremos por decidir que el dibujo se va a realizar sobre una superficie de ocho por ocho puntos (cuatro por cuatro símbolos de mosaico) sobre la pantalla, y dibujaremos, por tanto, una sencilla rejilla como la que se muestra en la figura 2.13. Ahora realizaremos la forma del hombrecillo, rellenando un patrón de bloques del interior de la matriz de ocho por ocho.

CODIGOS DE CARACTERES.



128+133+143+128

132+140+142+140

128+132+142+128

128+138+128+138

Fig. 2.13. Mosaico de puntos para producir la forma elemental de un hombrecillo.

Una vez realizado el patrón sobre la rejilla, tenemos que cambiarlo a códigos de carácter para poder imprimirlo en la pantalla. Recuerde que cada carácter consiste en cuatro elementos de bloque formando una matriz de 2x2, por tanto, podemos ya marcar los espacios de caracteres en nuestra rejilla. A continuación podemos

empezar con las dos primeras filas del patrón y convertir los patrones de bloque en cuatro códigos de símbolo, como los que se muestran a la derecha de la rejilla. El proceso se repite para las otras filas del dibujo, y, por tanto, tenemos un total de dieciséis códigos de carácter en cuatro grupos de cuatro.

Los códigos de carácter se guardan ahora en una matriz variable de 4x4 que llamamos a. Todo esto lo establecemos utilizando un bucle de lectura READ y un conjunto de instrucciones de datos (DATA). Una vez fijada la matriz, tenemos que decidir en qué parte de la pantalla queremos que esté el hombre. Si queremos que el hombre esté en la fila 1Ø y la columna 15, sólo tenemos que fijar las variables r y c en 10 y 15, respectivamente. Observe que estas coordenadas fijan el ángulo superior izquierdo del patrón que queremos pintar.

Una vez elegida la posición, el hombre puede imprimirse simplemente utilizando dos bucles (desde 1 a 4), que van llamando a los elementos de la matriz según los van necesitando. Las variables de estos bucles son i (para posiciones de columnas), y j (para las filas).

Para saber las posiciones de fila y columna para la instrucción PRINT AT, añadimos j á r, é i á c. Como los valores de i y j comienzan en 1, para tener la posición correcta en la pantalla deberá restarse 1 a los valores r+j, c+i. Observe el punto y coma al final de la línea PRINT AT que permite imprimir el símbolo siguiente en el espacio contiguo. El programa final para imprimir nuestro monigote es el que se muestra en la figura 2.14.

```
100 REM La figura de un hombre utilizando
graficos de mosaico.
120 DIM a(4,4)
125 REM Da valores a los simbolos de mosaico.
130 FOR j=1 TO 4
140 FOR i=1 TO 4
150 READ a(i,j)
160 NEXT i
170 NEXT j
180 DATA 128,133,143,128
190 DATA 132,140,142,140
200 DATA 128,132,142,128
210 DATA 128,138,128,138
215 REM Dibuja el hombre en la pantalla.
220 LET r=10
230 LET c=15
240 FOR j=1 TO 4
250 FOR i=1 TO 4
260 PRINT AT r+j-1,c+i-1;CHR$ a(i,j);
270 NEXT i
280 NEXT j
290 STOP
```

Fig. 2.14. Programa para dibujar un "monigote" utilizando gráficos de mosaico.

Si podemos dibujar un monigote, evidentemente podemos dibujar mucho más por toda la pantalla. Lo hemos hecho con el programa de la figura 2.15., comenzando con el primer hombre en la posición Ø,Ø, en la parte superior izquierda de la pantalla.

```
100 REM Varias figuras de hombre
110 REM utilizando graficos de mosaico.
120 DIM a(4,4)
125 REM Da valores a los simbolos de mosaico.
130 FOR j=1 TO 4
140 FOR i=1 TO 4
150 READ a(i,j)
160 NEXT i
170 NEXT j
180 DATA 128,133,143,128
190 DATA 132,140,142,140
200 DATA 128,132,142,128
210 DATA 128,138,128,138
215 REM Dibuja  los hombres en la pantalla.
220 FOR r=0 TO 16 STEP 5
230 FOR c=0 TO 28 STEP 4
240 FOR j=1 TO 4
250 FOR i=1 TO 4
260 PRINT AT r+j-1,c+i-1;CHR$ a(i,j);
270 NEXT i
280 NEXT j
290 NEXT c
300 NEXT r
310 STOP
```

Fig. 2.15. Programa que dibuja múltiples monigotes por toda la pantalla.

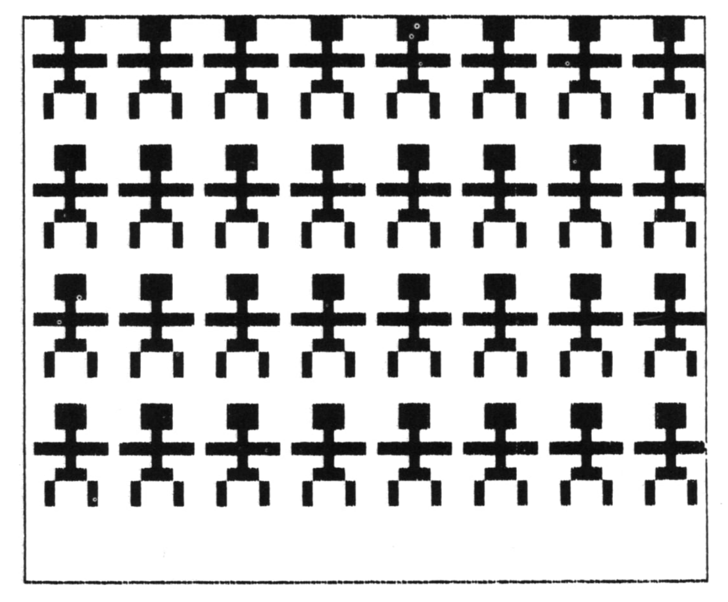


Fig. 2.16. Imagen que produce el programa de la figura 2.15.

Un punto a notar es que, cada vez que se ha dibujado un hombre, c se incrementa en 4 y, después de toda una hilera de hombres, r se incrementa en 5, para dejar una línea en blanco sobre la pantalla entre cada hilera de hombres.

Evidentemente podemos dibujar el monigote en posiciones aleatorias en la pantalla, utilizando la instrucción PRINT AT, pero, si lo hacemos, el máximo valor de c debe limitarse a 28, y el máximo valor de r a 16, para prevenir que los valores de c y r se salgan de los límites de pantalla, produciendo un error. En este caso, c y r son las posiciones de columna y fila del carácter superior izquierdo del grupo que se usó para realizar la figura.

Capítulo 3

Gráficos de alta resolución

En la realización de dibujos serios, el modo gráfico de baja resolución del Spectrum no es suficientemente bueno, y, evidentemente, necesitaremos movernos a un nivel muy superior en la resolución de gráficos. Ya hemos explicado en el Capítulo Uno, que el modo de alta resolución permite controlar el estado de cada punto individual de cada espacio de carácter. El espacio de un carácter, en la pantalla del Spectrum, es una matriz de 8 × 8 puntos; y como hay 32 carácteres en cada fila, tendremos 256 puntos en horizontal. En la dirección vertical, las dos últimas filas de texto están reservadas para la visualización de los mensajes, de modo que la pantalla de alta resolución se ve limitada a 22 filas de texto que dan 176 puntos en dirección vertical. Así pues, la pantalla de visualización de alta resolución es de 256 × 176 puntos.

En la memoria del ordenador, el patrón de los estados de ocho puntos adyacentes de una fila se almacena como una palabra de 8 bit, y, por tanto, la memoria total para la alta resolución es de 256 × 176 dividido por ocho, o 5632 bytes de memoria. Con este esquema, cada punto puede estar "on" o "off" (activado o sin activar), o en otras palabras, está en color de tinta INK (on) o en color de papel PAPER (off). Los colores se establecen mediante la instrucción INK y PAPER, del mismo modo que si fueran texto normal o gráficos de mosaico.

La información del color se almacena en una zona diferente de la memoria, y se guarda un par de colores por cada espacio de carácter de la pantalla.

Esto puede presentar algún problema, como veremos más adelante, pero permite la utilización de todos los colores en la pantalla al mismo tiempo. En muchos otros ordenadores, la información del color se almacena por cada PIXEL. Se necesitan dos bits de datos para cada pixel para cuatro colores, y tres bits para ocho colores, de modo que la memoria necesaria se duplica o triplica con respecto a

la necesaria para visualizar los dibujos en blanco y negro. Para reducir la cantidad de memoria necesaria para la visualización, el número de colores que se puede utilizar a un tiempo se reduce con frecuencia a dos o a cuatro. Los ordenadores que utilizan esta técnica permiten que dos puntos adyacentes sean de diferente color, cosa que no siempre es posible con el Spectrum, como veremos más adelante.

Activación y desactivación de puntos individuales

Con el modo de alta resolución es muy sencillo escribir un punto individual en la pantalla y darle un color determinado. Para ello, la instrucción adecuada es PLOT, y la palabra de la instrucción se puede teclear con la tecla Q cuando el cursor parpadeante presente la letra K. El comando completo es la forma:

```
100 PLOT x,y
```

donde x é y son las coordenadas del punto que se va a escribir. El punto elegido de la pantalla se escribirá con el color de tinta INK vigente en ese momento. En la pantalla de alta resolución tenemos muchos más puntos de los que teníamos en la pantalla de baja

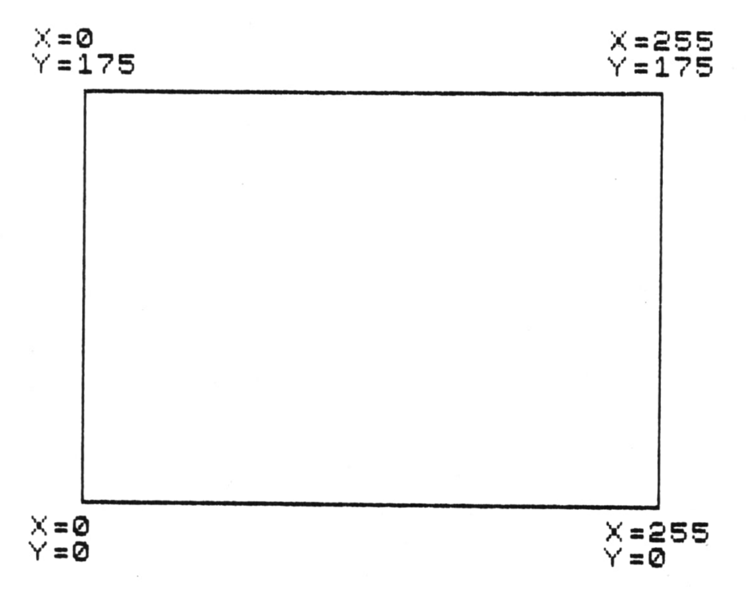


Fig. 3.1. visualización del esquema de la pantalla para gráficos de alta resolución.

resolución, y, por tanto, los valores de x é y pueden ser mucho mayores. De hecho, para el modo de alta resolución, las coordenadas x é y se mueven sobre una rejilla de 256 × 176 puntos. De este modo, la x tiene una gama de valores que va desde 0 a 255, y la gama para la y es de 0 a 175. El valor 0 de x se encuentra en la parte izquierda, y se va incrementando hacia el borde derecho de la pantalla, que es 255. A diferencia de la pantalla de texto, el valor de y es 0 en la esquina inferior izquierda de la pantalla, y aumenta hasta 175 hacia arriba de la pantalla. Así pues, el ángulo superior izquierdo es el punto 0,175, y el ángulo inferior derecho es el 255,0. El aspecto general se muestra en la figura 3.1.

Pruebe el programa listado en la Figura 3.2. Este programa selecciona aleatoriamente valores de x é y, y dibuja puntos en posiciones también aleatorias por toda la pantalla.

```
100 REM Puntos aleatorios.
110 FOR n=1 TO 1000
120 LET x=INT (RND*256)
130 LET y=INT (RND*176)
140 PLOT x,y
150 NEXT n
```

Fig. 3.2. Programa que produce puntos aleatoriamente, utilizando la técnica de alta resolución.

Igual que en modo de texto, el Spectrum se mueve por la pantalla cambiando de posición mediante un cursor. El cursor de texto se visualiza normalmente como un espacio iluminado, pero el cursor gráfico no se visualiza, aunque el ordenador mantiene en la memoria su posición x,y en cada momento. En el instante de ejecutar una operación de dibujo, la posición del cursor que estaba almacenada en la memoria, se actualiza inmediatamente.

Cuando se conecta el Spectrum, después de limpiar la pantalla con la instrucción CLS, el cursor se coloca automáticamente en una posición x,y = 0,0. Cuando ejecutamos una instrucción PLOT, la posición del cursor se establece directamente en el valor x,y especificado en la instrucción PLOT. De este modo, utilizando la instrucción PLOT podemos colocar el cursor en cualquier lugar de la pantalla. Casi siempre, en todos los dibujos de alta resolución, la primera instrucción es PLOT para posicionar el cursor.

Para reubicar un punto en la pantalla, se utiliza la instrucción INVERSE 1 antes de la instrucción PLOT. El comando INVERSE lo que hace es cambiar los colores INK y PAPER (uno por otro), de

forma que el punto está ahora escrito en el color de papel PAPER que esté en ese momento establecido para ese punto. Como el punto tiene ahora el color PAPER, (de fondo) desaparece. El resto de los puntos no se ve afectado. Después de realizar un comando, o una serie de comandos PLOT, puede usarse el comando INVERSEØ para volver al estado normal. La operación INVERSE puede incluirse en la instrucción PLOT del modo siguiente:

150 PLOT INVERSE1;x,y:INVERSEØ

Volveremos a estudiar mejor este comando INVERSE en el Capítulo Seis.

Selección del color para los gráficos

Al inicializar el Spectrum, éste trabajará sobre un fondo blanco y el texto negro. Las líneas y puntos de alta resolución también se visualizarán de color negro sobre un fondo blanco. El color del dibujo se controla, en alta resolución, con el comando INK, exactamente del mismo modo que en la visualización de textos. El color de fondo, PAPER, no se ve afectado por el comando PLOT, y, por tanto, el punto aparecerá contrastado contra cualquier color PAPER que ya existiera alrededor de ese punto, en la pantalla. Trate de realizar el programa que se lista en la figura 3.3, que dibuja puntos sobre la pantalla en posiciones aleatorias y con un color INK (de texto) elegido también aleatoriamente para cada punto.

```
100 REM Puntos aleatorios coloreados.
110 FOR n=1 TO 10000
120 LET i=INT (RND*7)
130 LET x=RND*255
140 LET y=RND*175
150 INK i
160 PLOT x,y
170 NEXT n
```

Fig. 3.3. Programa que produce puntos coloreados de forma aleatoria.

Según va avanzando el dibujo, usted advertirá que al principio van apareciendo puntos individualmente, pero, más adelante, grupos de puntos cambian de color todos a la vez. Finalmente, usted verá que los colores se asignan por igual a todos los puntos que pertenezcan al mismo espacio de carácter de la pantalla.

Se debe a que el Spectrum almacena la información del color separadamente de la del punto. La información de color se almacena en términos de posiciones de caracteres de texto, y el Spectrum sólo permite un par de colores para cada espacio de carácter de texto en la pantalla. De este modo, cuando se dibuja un nuevo punto, el color INK del espacio de carácter en el que se halla alojado se activa con el color INK actual. Si existen algunos otros puntos alojados en este mismo espacio de carácter, cambiarán todos de color, tomando el del nuevo punto. Evidentemente, esto es una limitación para la creación de dibujos con el Spectrum, ya que los puntos adyacentes no pueden dibujarse con colores diferentes.

Combinación de textos y gráficos

En algunos ordenadores personales, el modo de texto y el modo de alta resolución tienen sistemas de visualización diferentes, y puede ser muy difícil combinar texto y gráficos en una sola imagen sobre la pantalla. Como el Spectrum almacena los símbolos del texto como conjuntos de puntos en la memoria de pantalla, no hay dificultad alguna para mezclar texto y gráficos sobre la pantalla. Los símbolos de texto se producen simplemente utilizando PRINT o PRINT AT, y los puntos gráficos pueden dibujarse sobre los símbolos de texto, usando los comandos PLOT.

Es importante tener en cuenta, cuando se mezcla texto con gráficos, que mientras que en la pantalla de texto las filas se numeraban desde la parte superior a la inferior, con la fila 0 en la parte superior, cuando se utilicen gráficos, la coordenada y comienza (y vale 0) en la parte inferior de la pantalla, incrementándose según vamos subiendo por la pantalla. Como cada símbolo tiene una anchura de ocho puntos, y una altura de ocho, los valores de fila y columna se convierten fácilmente en valores de x é y, simplemente multiplicándolos por ocho. Esto es, para el cálculo de la x utilizaremos la ecuación:

$$x \times 8 * c$$

Para calcular la y necesitamos corregir la diferencia de dirección entre los valores de r é y. La ecuación es la siguiente:

$$y = 175 - 8 * r$$

Para convertir los valores x é y en r y c, no tenemos más que dividir x é y por 8, y luego redondearlos tomando su parte entera

con el comando INT. Observe que la instrucción INT sólo elimina la parte fraccionaria de un número. De nuevo debe hacerse una corrección para las r é y que se presentaban de modo diferente. Los cálculos son los siguientes:

```
r=INT((175-y)/8)
c=INT (x/8)
```

Un punto a notar es que, a diferencia del comando PRINT AT, el comando PLOT lleva primero la coordenada horizontal (x).

Dibujo de líneas

Dibujar puntos es interesante, pero en la mayoría de los casos lo que desearemos es dibujar líneas en la pantalla. Mientras que en el modo de baja resolución teníamos que hacerlo calculando los puntos que debían dibujarse y después trazándolas, en el modo de alta resolución es mucho más sencillo, porque hay un comando especial para el dibujo de líneas. Este comando se llama DRAW y se obtiene pulsando la tecla W cuando el Spectrum está con el cursor en K. La forma de esta instrucción es:

```
100 DRAW x,y
```

Hay una diferencia muy importante entre los valores x,y utilizados por el comando DRAW, y los utilizados con el comando PLOT. En el caso de PLOT, los valores x é y especifican la posición que tienen esos puntos en la pantalla. En el caso de la instrucción DRAW, los valores de x é y están relacionados con la posición del cursor en la pantalla. Como ejemplo, supongamos que utilizamos el comando:

```
100 PLOT 30,20
```

Esta instrucción coloca el cursor en la posición x = 30, y = 20 de la pantalla, y, una vez allí coloca un punto en esa posición.

Si ahora utilizamos el comando

```
110 DRAW 30,20
```

se dibujará una línea desde el punto que acabamos de dibujar a un nuevo punto 30 unidades a la derecha y 20 unidades hacia arriba. El

cursor se moverá luego al final de la línea que acabamos de dibujar. Esto quiere decir que el cursor se encuentra en la posición correspondiente a los valores x = 60 (30 + 30) é y = 40 (20 + 20).

Este método de cáculo de posiciones en la pantalla se llama dibujo relativo, y puede simplificar mucho el trazado de formas, ya que no tendremos que calcular las coordenadas absolutas x é y para cada punto de la figura que vamos a dibujar.

Un punto a hacer notar es que en la instrucción DRAW, ambas, x é y pueden ser negativas. Un valor de x negativo significa simplemente que la línea se dibuja hacia la izquierda de la posición actual, y una negativa significa que la línea se traza hacia abajo desde el punto en que nos encontramos.

Igual que en el caso del trazado de puntos individuales con el comando PLOT, las líneas producidas utilizando el comando DRAW, se dibujan con el color INK vigente.

Dibujo de líneas entre puntos especificados

En muchos casos desearemos ser capaces de dibujar una línea entre dos puntos específicos de la pantalla. Supongamos que esos puntos se indican con las coordenadas x1 e y1 y x2 e y2. Puesto que la instrucción DRAW utiliza coordenadas relativas, necesitaremos ciertos cálculos para obtener los parámetros x é y de la instrucción DRAW.

El primer paso para el dibujo de la línea es colocar el cursor en uno de los finales de línea, por ejemplo en el punto x1,y1. Esto se realiza fácilmente utilizando PLOT x1,y1 para dibujar un punto en el lugar elegido. Después deberemos calcular los valores de x é y para la instrucción DRAW. Para ello debemos encontrar la diferencia entre x1 y x2, y entre y1 é y2, esto es, x = x2—x1 é Y = y2—y1. No es necesario calcular los valores de x é y separadamente, ya que los podemos hacer con la instrucción DRAW, sustituyendo los términos x é y por las expresiones siguientes:

```
1ØØ DRAW x2—x1,y2—y1
```

En esta instrucción, el valor de x se hace negativo cuando x2 está a la izquierda de x1, y el término y se hace negativo si y2 se encuentra en la pantalla debajo de y1.

Después de haber dibujado una línea, el cursor se habrá movido

al final de la línea (x2,y2), en este caso. Si ahora queremos dibujar otra línea que comience desde el final de la anterior, no necesitaremos la instrucción PLOT, ya que el cursor se encontrará en la posición correcta.

Dibujando cintas

Ahora que ya podemos dibujar líneas y controlar el color de visualización de los elementos, vamos a experimentar y dibujar sobre la pantalla otras figuras. Comencemos generando una línea que se mueve cambiando de color.

El principio que rige este tipo de figuras es la elección previa de dos puntos aleatorios (x1,y1 y x2,y2) en la pantalla, para dibujar una línea entre ellos. Después alteraremos ligeramente la posición de los dos puntos y dibujaremos otra línea. El dibujo se va creando conforme se van dibujando más líneas con pequeños cambios en los valores x1 é y1, y x2 é y2.

Si realizamos el mismo cambio en los valores de los dos finales de la línea, su longitud permanecerá constante, y según se vayan dibujando las sucesivas líneas, se irá produciendo una cinta de color. Si los cambios que se realizan en los dos finales de la línea son diferentes, la anchura de la cinta cambia y ésta se retuerce en su movimiento por la pantalla.

Cuando el ordenador está calculando las coordenadas x e y, puede suceder que los valores de x e y que se obtengan caigan fuera de la gama de valores permitidos. Algunos tipos de ordenadores pueden tolerar esta situación, y llevan a cabo una operación de "enrrollamiento de la pantalla". La operación consiste en que un punto que se salga de la frontera derecha de la pantalla se corrige y reinserta a la izquierda de la pantalla. El Spectrum, sin embargo, cada vez que x ó y se salgan de los valores permitidos, únicamente parará el programa y dará una señal de error. Para evitar esta posible condición de error, necesitaremos probar los valores calculados para x e y antes de utilizarlos en el dibujo de líneas. Esto puede hacerse comprobando sencillamente si x es mayor que 255 o inferior a 0, y si y está entre 0 y 175.

En el programa de dibujo de figuras, cuando uno de los puntos alcanza el borde de la pantalla, el incremento para la x se cambia de signo, y las coordenadas se vuelvan a recalcular. Esta acción tiene el efecto de enviar la línea hacia atrás por la pantalla, como si se hubiera reflejado en el borde de ésta.

```
100 REM Patron de una linea que se mueve.
110 FOR n=1 TO 20
120 LET dx1=2-INT (5*RND)
130 LET dx2=2-INT (5*RND)
140 LET dy1=2-INT (5*RND)
150 LET dy2=2-INT (5*RND)
160 LET x1=20+RND*200
170 LET y1=20+RND*130
180 LET X2=20+RND*200
190 LET y2=20+RND*130
200 INK INT (7*RND)
210 PAPER 7
220 CLS
230 FOR k=1 TO 500
240 PLOT x1,y1
250 DRAW x2-x1,y2-y1
260 LET x1=x1+dx1
270 IF x1<=255 AND x1>=0 THEN GO TO 310
280 LET dx1=-dx1
290 LET x1=x1+dx1
300 INK INT (RND*7)
310 LET x2=x2+dx2
320 IF x2<=255 AND x2>=0 THEN GO TO 360
330 LET dx2=-dx2
340 LET x2=x2+dx2
350 INK INT (RND*7)
360 LET y1=y1+dy1
370 IF y1<=175 AND y1>=0 THEN GO TO 410
380 LET dy1=-dy1
390 LET y1=y1+dy1
400 INK INT (RND*7)
410 LET y2=y2+dy2
420 IF y2<=175 AND y2>=0 THEN GO TO 460
430 LET dy2=-dy2
440 LET y2=y2+dy2
450 INK INT (RND*7)
460 NEXT k
470 NEXT n
```

Fig. 3.4. Programa que produce el dibujo de una cinta.

Un programa para dibujar una forma de este tipo se lista en la Figura 3.4. Para que el dibujo tenga más colorido, cada vez que la línea alcanza uno de los bordes de la pantalla, cambiará de color.

Este programa también demuestra las limitaciones que tiene el método del Spectrum para almacenar la información de los colores INK y PAPER. Como el color controla un espacio de símbolo completo, si se utiliza un color INK diferente en un espacio donde ya había algún punto en color, todos los puntos cambiarán al nuevo color de tinta. Esto produce un efecto de escalonamiento, como si el dibujo se hubiera realizado sobre otro dibujo de diferente color. Aparte de esta limitación, el programa producirá diseños abstractos muy atractivos.

Producción de diseños de moiré

Otro tipo de diseño muy atractivo, utilizando gráficos de alta resolución, puede ser un diseño de moiré. Este tipo de diseños es fácil de hacer, eligiendo simplemente en la pantalla un punto de partida aleatorio, y luego dibujando un conjunto de líneas radiales desde ese punto a los bordes de la pantalla. Cambiando la inclinación de las líneas y la posición del origen, podemos generar una selección de diseños que se asemejan a los que se ven a veces en la seda o en el tafetán. El programa listado en la Figura 3.5 utiliza funciones aleatorias para generar los orígenes de los diseños y su pendiente. La figura 3.6. muestra un patrón típico, como el que puede verse en la pantalla.

```
100 REM Patrones de "moire".
110 INK 0: PAPER 7
120 FOR k=1 TO 100
130 CLS
140 LET cx=20+200*RND
150 LET cy=20+130*RND
160 LET s=2+INT (3*RND)
170 FOR x=0 TO 255 STEP s
180 PLOT cx,cy
190 DRAW x-cx,0-cy
200 PLOT cx,cy
210 DRAW x-cx,175-cy
220 NEXT x
230 FOR y=0 TO 175 STEP s
240 PLOT cx,cy
250 DRAW 0-cx,y-cy
260 PLOT cx,cy
270 DRAW 255-cx,y-cy
280 NEXT y
290 PAUSE 500
300 NEXT k
```

Fig. 3.5. Programa que produce un sencillo diseño de moiré.

Podemos producir programas con mucho más colorido, haciendo que los colores de texto (INK) y de fondo (PAPER) se activen de un modo aleatorio, antes de dibujar el diseño, como se hace en el programa listado en la Figura 3.7. Con un comando CLS después del comando PAPER, dejaremos la pantalla completa con el color PAPER elegido. El diseño lo realizaremos ahora en la parte superior y con el color de texto (INK) que deseemos. En este programa, para cada nuevo diseño se ha elegido un conjunto aleatorio de colores (INK, PAPER y BORDER-para el borde).

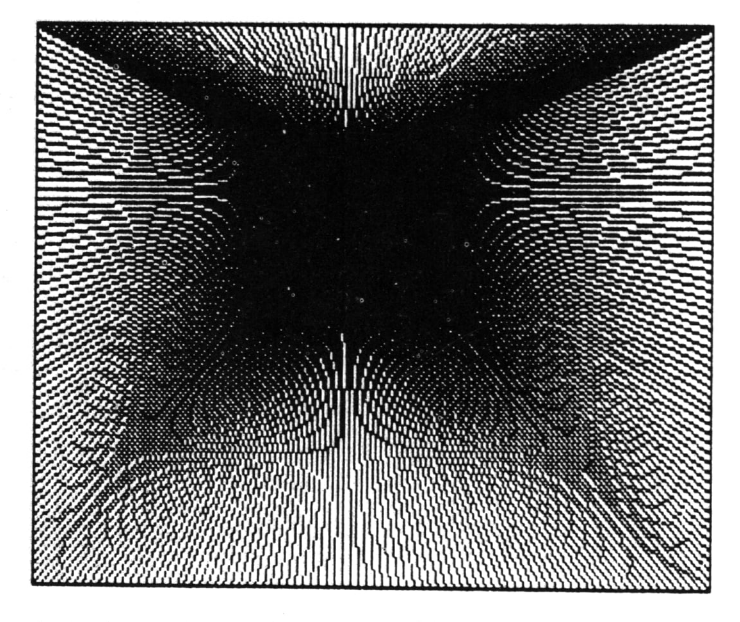


Fig. 3.6. Dibujo producido por el programa moiré.

```
100 REM Patrones de "moire" con color.
110 FOR k=1 TO 100
120 LET cx=20+200*RND
130 LET cy=20+130*RND
140 LET p=INT (RND*8)
150 LET i=INT (RND*8)
160 IF p=i THEN GO TO 150
170 PAPER p
180 INK i
190 BORDER INT (RND*8)
200 CLS
210 LET s=2+INT (3*RND)
220 FOR x=0 TO 255 STEP s
230 PLOT cx,cy
240 DRAW x-cx,0-cy
250 PLOT cx,cy
260 DRAW x-cx,175-cy
270 NEXT x
280 FOR y=0 TO 175 STEP s
290 PLOT cx,cy
300 DRAW 0-cx,y-cy
310 PLOT cx,cy
320 DRAW 255-cx,y-cy
330 NEXT y
340 PAUSE 500
350 NEXT k
```

Fig. 3.7. Programa para realizar patrones de moiré con colores.

Dibujo de línea de puntos

El comando para el dibujo de líneas dibuja una línea continua entre los puntos x1,y1, y x2,y2.

Supongamos, sin embargo, que deseamos una línea de puntos entre los dos puntos dados. El Spectrum no tiene un comando apropiado para ello, por tanto debemos crear una rutina de dibujo de líneas, que calcule y dibuje puntos individuales, para construir la línea.

La línea de puntos se dibuja calculando los pixels de la línea que deben activarse (iluminarse) e iluminarlos con el comando PLOT. La rutina básica de dibujo de líneas calcula la diferencia entre x1 y x2, y entre y1 e y2, y toma el mayor valor de estas diferencias como número de puntos a dibujar.

El siguiente paso es calcular los incrementos de x e y entre dos puntos sucesivos. Si el número mayor de puntos está en la dirección de eje x, se fija 1 como incremento para las x. El incremento para las y será una fracción calculada dividiendo la diferencia y2—y1 por el número total de puntos np. Si es la dirección y, la que tiene el mayor número de puntos, será este incremento el que tomará valor 1 y el de las x tomará un valor inferior a uno.

Para obtener una línea de puntos, necesitamos dibujar puntos alternadamente, a todo lo largo de la línea, por tanto el bucle de diseño avanza de dos en dos unidades. Al final de la línea, el bucle de dibujo traza un punto único adicional en el punto x2,y2 para asegurarse de que la línea tiene la longitud correcta. En el programa que se muestra en la Figura 3.8., la rutina de dibujo de líneas se ha convertido en subrutina, y el programa principal dibuja una serie de líneas de puntos entre puntos aleatorios de la pantalla. Deberá tenerse en cuenta que los valores de x1 y1 y x2 y2, que representan el comienzo y final de la línea, deben establecerse en el programa principal, antes de llamar a la subrutina.

Modificando la subrutina de dibujo, también pueden dibujarse líneas de guiones, o incluso con guiones y puntos alternadamente. Para obtener una línea de guiones, la rutina se saltará tres puntos cada vez, pero, en este caso, dibujará dos puntos por cada paso por la subrutina de dibujo.

Le dejamos este caso para sus experimentaciones.

```
100 REM Lineas de puntos.
110 FOR n=1 TO 20
120 LET x1=INT (RND*255)
```

```
130 LET x2=INT (RND*255)
140 LET y1=INT (RND*175)
150 LET y2=INT (RND*175)
160 LET xs=1
170 LET ys=1
180 LET xi=1
190 LET yi=1
200 LET dx=x2-x1
210 LET dy=y2-y1
220 IF dx<0 THEN LET xs=-1
230 IF dy<0 THEN LET ys=-1
240 LET nx=ABS (dx)
250 LET ny=ABS (dy)
260  IF nx>=ny THEN LET np=nx: LET yi=ny/nx
270 IF ny>nx THEN LET np=ny: LET xi=nx/ny
280 PLOT x1,y1
290 LET s=INT (RND*3)+2
300 FOR j=0 TO np STEP s
310 LET x=x1+xs*INT (j*xi+.5)
320 LET y=y1+ys*INT (j*yi+.5)
330 PLOT x,y
340 NEXT j
350 PLOT x2,y2
360 NEXT n
```

Fig. 3.8. Programa que dibuja líneas de puntos.

Dibujos de triángulos

El triángulo, con sus tres lados y su tres esquinas, es quizá la figura o forma más sencilla de dibujar. Para ello, necesitaremos conocer la posición en la pantalla de las tres esquinas, que llamaremos x1,y1,x2,y2 y x3,y3, como puede verse en la figura 3.9. El proceso de dibujo de un triángulo incluye el trazado de una línea de x1,y1 a x2,y2 luego una segunda línea x2,y2 a x3,y3. Finalmente, el triángulo se completa dibujando la línea desde x3,y3 a x1,y1.

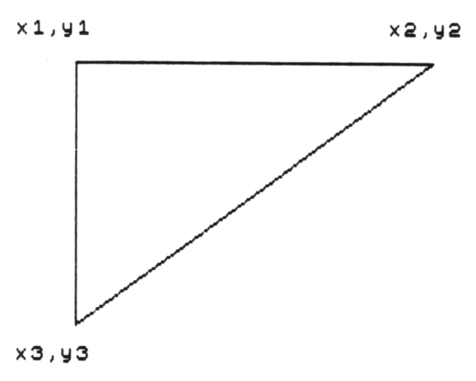


Fig. 3.9. Coordenadas de dibujo de un triángulo

Si empezamos el proceso de dibujar el triángulo, el primer paso es posicionar el cursor en una de las esquinas, por ejemplo el punto x1,y1. Una vez allí, colocaremos un punto mediante el comando PLOT, que posicionará el cursor y lo dejará preparado para el trazado de la primera línea. Los tres lados del triángulo se dibujan utilizando tres comandos DRAW. La Figura 3.10 muestra un programa sencillo que selecciona aleatoriamente un conjunto de tres puntos, para luego dibujar con ellos triángulos. En este caso, la operación de dibujo de los triángulos se ha realizado como subrutina.

```
100 REM Programa de dibujo aleatorio de
triangulos.
110 FOR s=1 TO 20
120 BORDER INT (RND*8)
130 LET p=INT (RND*8)
140 PAPER p: CLS
150 FOR n=1 TO 15
160 REM Establece los vertices
170 LET x1=INT (RND*255)
180 LET y1=INT (RND*175)
190 LET x2=INT (RND*255)
200 LET y2=INT (RND*175)
210 LET x3=INT (RND*255)
220 LET y3=INT (RND*175)
230 REM Establece el color del dibujo.
240 LET c=INT (RND*8)
250 IF c=p THEN GO TO 240
260 INK c
270 REM Dibuja el triangulo.
280 PLOT x1,y1
290 DRAW x2-x1,y2-y1
300 DRAW x3-x2,y3-y2
310 DRAW x1-x3,y1-y3
320 NEXT n
330 PAUSE 200
340 NEXT s
350 INK 0
```

Fig. 3.10. Programa que dibuja un triangulo aleatorio.

Imágenes especulares

La producción de triángulos puede dar patrones interesantes, pero podemos obtener resultados más atractivos utilizando el principio de las imágenes especulares. Colocaremos cuatro patrones de imágenes especulares alrededor del centro de la pantalla, de forma que cada patrón llene un cuarto de pantalla.

La técnica a utilizar consiste en dibujar un triángulo de forma aleatoria en el cuadrante superior derecho de la pantalla, añadiendo las coordenadas x,y del primer punto del triángulos a las coordenadas x,y del centro de la pantalla, y dibujando luego el triángulo con tres comandos DRAW, como en el último programa. El mismo triángulo básico vuelve a dibujarse, pero esta vez la coordenada y de su primer punto se sustrae de las coordenadas del centro de la pantalla, de forma que el triángulo se dibuja debajo de la línea media de la pantalla. Para invertir el triángulo, se transponen las coordenadas y de los tres comandos DRAW, de forma que y2-y1 es ahora y1-y2, y así sucesivamente. En lo que respecta a los otros dos cuadrantes de la pantalla, se utilizan las mismas dos secuencias, pero, en este caso, la x del primer punto se sustrae de la coordenada x del centro, y los términos x se transponen en los tres comandos DRAW, para colocar los triángulos a la izquierda del centro y girarlos de izquierda a derecha.

La Figura 3.11 ofrece el listado de un programa que realiza este tipo de patrón que se asemeja mucho a los dibujos de un caleidoscopio. En el programa, se han utilizado colores aleatorios para las series de triángulos que forman el patrón. Los sucesivos patrones se producen con colores aleatorios para el fondo y los bordes.

El resultado en la pantalla es similar al que se muestra en la Figura 3.12, pero llena de colorido.

```
100 REM Programa del kaleidoscopio.
110 FOR s=1 TO 20
120 BORDER INT (RND*8)
130 LET p=INT (RND*8)
140 PAPER p: CLS
150 FOR n=1 TO 20
160 REM Establece los vertices.
170 LET x1=(RND*127)
180 LET y1=(RND*87)
190 LET x2=(RND*127)
200 LET y2=(RND*87)
210 LET x3=(RND*127)
220 LET y3=(RND*87)
230 REM Establece el color del dibujo.
240 LET c=INT (RND*8)
250 IF c=p THEN GO TO 240
260 INK c
270 REM Dibuja los triangulos.
280 PLOT 128+x1,88+y1
290 DRAW x2-x1,y2-y1
300 DRAW x3-x2,y3-y2
310 DRAW x1-x3,y1-y3
320 PLOT 128+x1,88-y1
330 DRAW x2-x1,y1-y2
```

```
340 DRAW x3-x2,y2-y3
350 DRAW x1-x3,y3-y1
360 PLOT 128-x1,88-y1
370 DRAW x1-x2,y1-y2
380 DRAW x2-x3,y2-y3
390 DRAW x3-x1,y3-y1
400 PLOT 128-x1,88+y1
410 DRAW x1-x2,y2-y1
420 DRAW x2-x3,y3-y2
430 DRAW x3-x1,y1-y3
440 NEXT n
450 PAUSE 200
460 NEXT s
470 INK 0
```

Fig. 3.11. Programa sencillo de caleidoscopio.

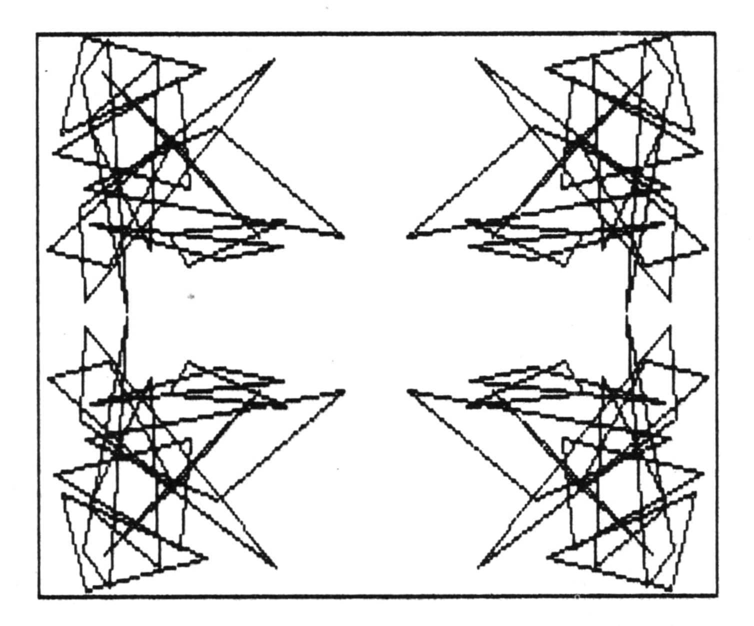


Fig. 3.12. Dibujo típico de caleidoscopio.

Dibujos de rectángulos y cuadrados

Dibujemos ahora figuras con cuatro lados y cuatro vértices, como los rectángulos. El método más sencillo consiste en calcular las coordenadas de los cuatro vértices del rectángulo para luego trazar cuatro líneas que unan los puntos formando los lados del rectángulo. Como ejemplo, podemos querer dibujar un rectángulo de 100 uni-

dades de anchura y 50 de alto, eligiendo una posición para el vértice inferior izquierdo (x1,y1) de 40,50. Los valores de los otros vértices se muestran en la Figura 3.13.

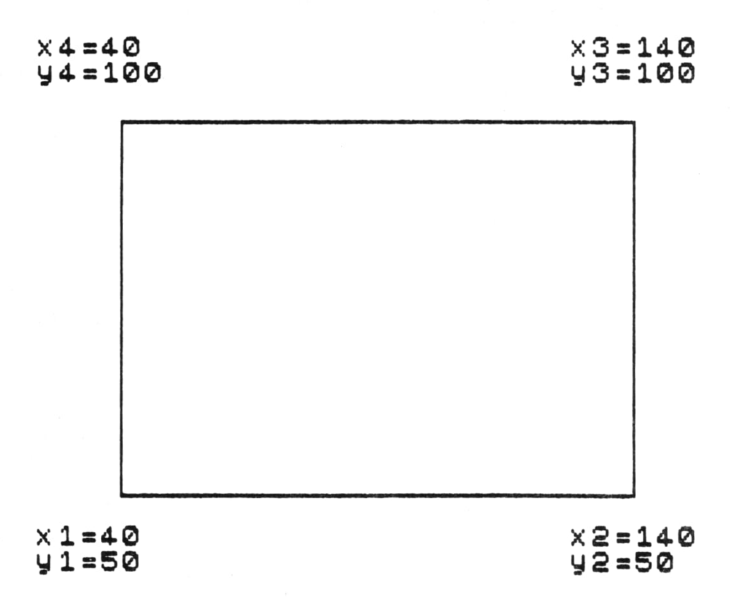


Fig. 3.13. Dibujo de las coordenadas de un rectángulo.

Todo lo que tenemos que hacer ahora es utilizar un comando PLOT para colocar el cursor en el vértice inferior izquierdo, y con cuatro líneas que forman el rectángulo. Este sistema se ilustra en la Figura 3.14.

```
100 REM Dibujo de un rectangulo
110 REM utilizando las coordenadas x e y de
sus vertices
120 LET x1=40
130 LET y1=50
140 LET x2=140
150 LET y2=50
160 LET x3=140
170 LET y3=100
180 LET x4=40
190 LET y4=100
200 PLOT x1,y1
210 DRAW x2-x1,y2-y1
220 DRAW x3-x2,y3-y2
230 DRAW x4-x3,y4-y3
240 DRAW x1-x4,y1-y4
```

Fig. 3.14. Programa que dibuja un rectángulo utilizando las coordenadas de sus vértices.

Utilizando la anchura y la altura

Dibujar un rectángulo utilizando las coordenadas de sus vértices no es el modo mejor de utilizar el comando DRAW del Spectrum. Los rectángulos se definen por su anchura w, y su altura h, como puede verse en la Figura 3.15. Si utilizamos estos datos, podemos aprovechar el método de coordenadas relativas que utiliza el comando DRAW. En este caso sólo necesitamos dar las coordenadas de un vértice, y los valores de la anchura y h de la altura del rectángulo.

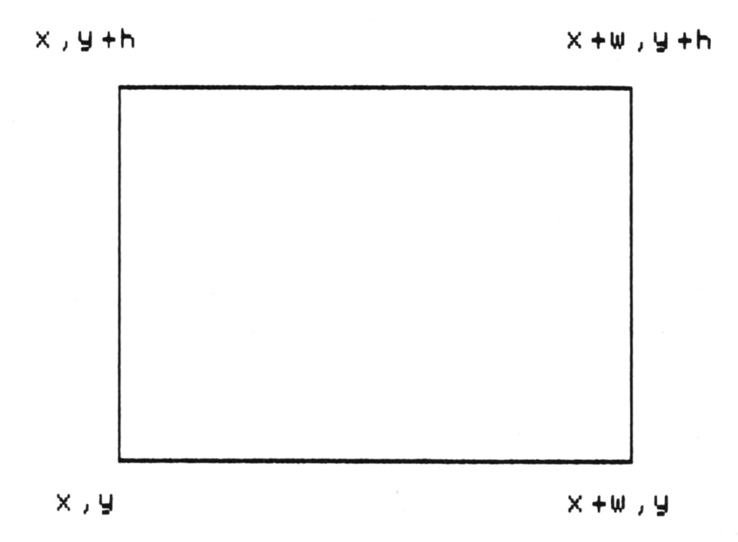


Fig. 3.15. Coordenadas de un rectángulo en función de su anchura y altura.

Utilizaremos una instrucción PLOT para colocar el punto y el cursor en el ángulo inferior izquierdo del rectángulo. La primera línea es horizontal y sigue el fondo del rectángulo, por tanto, el valor de la y del comando DRAW será 0. La longitud de esta línea es de w unidades, por tanto, la primera instrucción DRAW toma la forma:

```
11∅ DRAW w,∅
```

El lado superior del rectángulo vuelve a ser de w unidades de largo, pero está dibujado de derecha a izquierda, por lo tanto, la x debe ser negativa, es decir, —w, con el valor y = 0. La última línea es

vertical y, por tanto, el término x es 0 y el término y es —h, ya que la línea se traza hacia la parte inferior de la pantalla. Todos estos detalles se muestran en el programa listado en la Figura 3.16.

```
100 REM Dibujo de un rectangulo
110 REM utilizando la altura y la anchura
120 LET x=50
130 LET y=50
140 LET w=100
150 LET h=50
160 REM Dibuja el vertice inf. derecho.
170 PLOT x,y
180 REM Traza el lado inferior.
190 DRAW w,0
200 REM Traza el lado derecho.
210 DRAW 0,h
220 REM Traza el lado superior.
230 DRAW -w,0
240 REM Traza el lado izquierdo.
250 DRAW 0,-h
```

Fig. 3.16. Programa que dibuja un rectángulo, utilizando como datos su altura y anchura.

El cuadrado no es más que una versión especial del rectángulo, donde la altura, h y la anchura, w, son iguales. Para dibujar un cuadrado se puede utilizar la rutina básica para el rectángulo, fijando h = w, o bien otra rutina diferente utilizando una única variable w. En este caso, el término h del rectángulo se sustituye sencillamente por el término w.

Problemas con los límites de la pantalla

Si intentamos dibujar un rectángulo de 100 unidades de ancho, con su ángulo inferior en una posición x de 200, por ejemplo, el programa fallará y aparecerá en pantalla un mensaje indicando que el valor está fuera de límite. Esto se debe a que la coordenada x ha tomado un valor más allá del máximo permitido que está en 255. En algunos ordenadores personales, esta condición se salva automáticamente por la instrucción de dibujo de líneas, dibujando la línea hasta el final de la pantalla, o dibujando la parte de la línea que quedaría fuera de la pantalla a la izquierda de ésta. Recordemos, sin embargo, que el comando de dibujo de líneas del Spectrum no lleva incorporado un sistema para corregir esta situación.

```
100 REM Rectangulos aleatorios
110 REM con corte en el borde de la pantalla.
120 FOR s=1 TO 20
130 BORDER INT (8*RND)
140 LET p=INT (8*RND)
150 PAPER p
160 CLS
170 FOR n=1 TO 10
180 REM Establece la posicion del rectangulo
y su tamano.
190 LET x=INT (240*RND)
200 LET y=INT (165*RND)
210 LET w=10+INT (100*RND)
220 LET h=10+INT (100*RND)
230 LET c=INT (8*RND)
240 IF c=p THEN GO TO 230
250 INK c
260 REM Corta el rectangulo en el borde
de la pantalla.
270 IF x+w>255 THEN LET w=255-x
280 IF y+h>175 THEN LET h=175-y
290 REM Dibuja el rectangulo.
300 PLOT x,y
310 DRAW w,0
320 DRAW 0,h
330 DRAW -w,0
340 DRAW 0,-h
350 NEXT n
360 PAUSE 200
370 NEXT s
380 INK 0: PAPER 7
```

Fig. 3.17. Programa que dibuja rectángulos aleatorios con corrección de los valores que se salen fuera de la pantalla.

Si creamos una subrutina de dibujo de rectángulos, es fácil realizar ciertas pruebas que eviten que se produzcan valores fuera de los límites, causando errores. Antes de dibujar el primer lado del rectángulo, comprobaremos si x+w es mayor que 255, y, si es así, alteraremos el valor de w a w = 255—x. Con esto reducimos la anchura del rectángulo hasta que su borde derecho se encuentre en el límite derecho de la pantalla. Antes de dibujar el segundo lado del rectángulo, se lleva a cabo la misma prueba con y, pero, en este caso, el valor límite es 175. De nuevo, si el rectángulo se sale de la pantalla, se da a h un nuevo valor h = 175—y. Los lados superior e izquierdo del rectángulo se dibujan normalmente, utilizando los valores corregidos de w y h. El resultado que vemos en la pantalla es un rectángulo cortado por el borde de la pantalla en su lado derecho o en el superior, o en ambos. Para ver en la práctica todo lo que hemos estado estudiando, ejecute el programa de la figura 3.17, que producirá resultados similares a los de la figura 3.18.

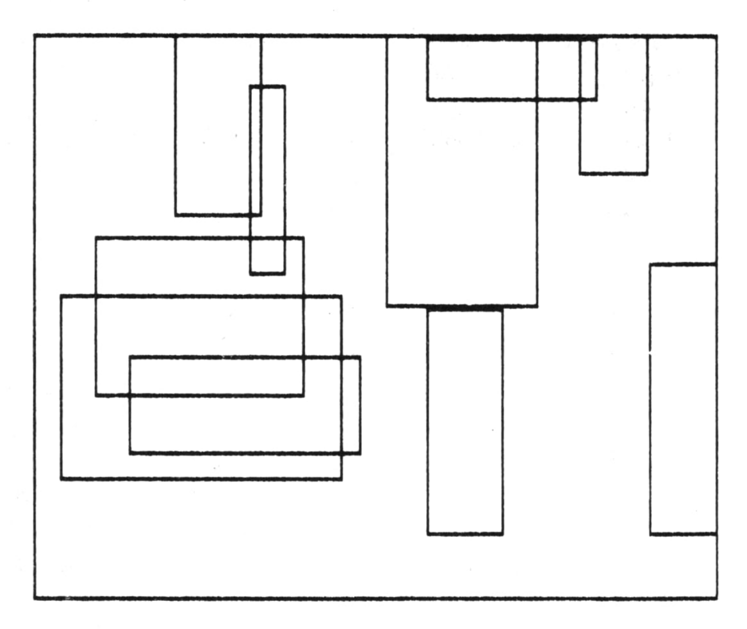


Fig. 3.18. Dibujo producido por un programa de trazado aleatorio de rectángulos.

Capítulo 4
Técnicas de dibujo

Hasta ahora hemos visto cómo se dibujan líneas y rectángulos, pero en muchos casos necesitaremos dibujar figuras más complejas, como, por ejemplo, polígonos, círculos y elipses. También podemos querer dibujar nuestras figuras formando un cierto ángulo con la horizontal. En este capítulo exploraremos las técnicas que se necesitan para realizar todas estas cosas, y comenzaremos dibujando círculos. De hecho, existen varios métodos para el dibujo de un círculo.

Utilización del comando CIRCLE (círculo)

La forma más fácil de dibujar un círculo en pantalla es utilizar el comando especial que tiene el Spectrum para el dibujo de círculos. Este comando se llama (de forma muy apropiada) CIRCLE, y la palabra clave del comando se obtiene presionando al mismo tiempo las teclas SYMBOL SHIFT y CAPS SHIFT, para obtener el cursor E. Luego se pulsarán las teclas SYMBOL SHIFT y H.

La instrucción CIRCLE tiene el siguiente formato:

```
100 CIRCLE x,y,r
```

donde x,y son las coordenadas del centro del círculo y r es el radio, medido en unidades de pantalla. Si deseáramos dibujar un círculo con un radio de 50 unidades colocado aproximadamente en el centro de la pantalla, el valor de x e y sería de 128 y 88, mientras que r = 50. Pruebe a teclear el comando directo:

```
CIRCLE   128,88,50
```

y obtendrá un círculo negro en el centro de la pantalla.

Para ver cómo actúa el comando CIRCLE, dentro de un programa, puede utilizar el que viene listado en la figura 4.1. En él, se

```
100 REM Dibujo de circulos
110 REM utilizando el comando CIRCLE
120 LET x=128
130 LET y=88
140 FOR r=10 TO 80 STEP 10
150 CIRCLE x,y,r
160 NEXT r
```

Fig. 4.1. Dibujo de círculos concéntricos utilizando el comando CIRCLE.

dibuja una serie de círculos concéntricos cuyo tamaño va aumentando para producir la imagen de la figura 4.2. Los valores de x e y son los mismos para todos los círculos, pero r se va incrementando en saltos, para dar sucesivamente círculos mayores.

Una limitación importante de la instrucción CIRCLE del Spectrum, es que no funciona si el círculo se sale de los límites de la pantalla.

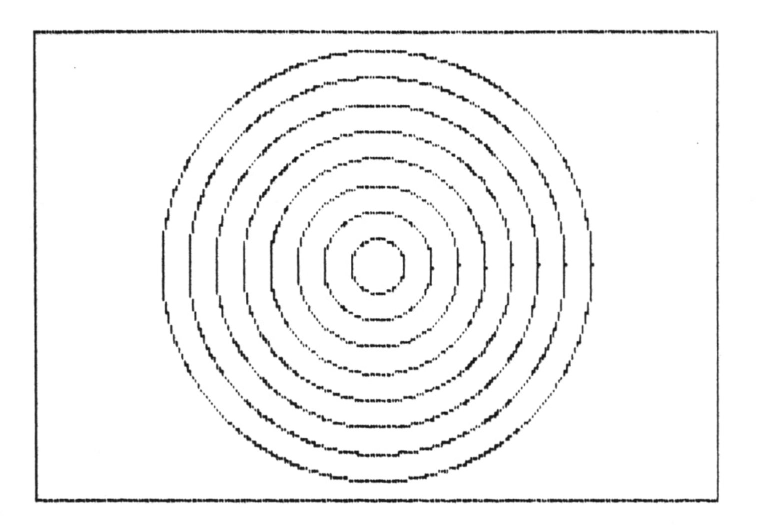


Fig. 4.2. Dibujo producido por el programa de la figura 4.1.

Si parte del círculo se sale de pantalla, el ordenador se parará e indicará que un valor está fuera de la gama permitida.

Supongamos que queremos dibujar una serie de círculos de tamaño aleatorio, por toda la pantalla, utilizando el comando CIRCLE. Para evitar problemas, debemos limitar los valores de x e y y r para cada círculo, con el fin de que no sobrepasen los bordes de la

pantalla. Si consideramos la x, su valor mínimo debe ser r para prevenir que se salga del borde izquierdo, y el máximo 255-r, para evitar que se salga por el lado derecho. En la función aleatoria utilizamos 255-2r para compensar el valor mínimo que asignamos a x, de forma que los cálculos finales de x son:

x = r + RND*(255 — 2*r)

Para el cálculo del valor de y, se utiliza una técnica similar, y el programa completo se muestra en la figura 4.3. El resultado producido en pantalla será similar al que se muestra en la figura 4.4.

```
100 REM Dibujo aleatorio de circulos
110 REM utilizando el comando CIRCLE.
120 FOR s=1 TO 10
130 CLS
140 REM Dibuja los bordes de la pantalla.
150 PLOT 0,0
160 DRAW 255,0
170 DRAW 0,175
180 DRAW -255,0
190 DRAW 0,-175
200 REM Dibuja los circulos.
210 FOR n=1 TO 15
220 LET r=5+RND*50
230 LET x=r+RND*(255-2*r)
240 LET y=r+RND*(175-2*r)
250 CIRCLE x,y,r
260 NEXT n
270 PAUSE 200
280 NEXT s
```

Fig. 4.3. Programa para dibujar círculos aleatoriamente.

Si los círculos se van dibujando y se prevee que van a rebasar el borde de la pantalla, será mejor utilizar otra de las técnicas de dibujo de círculos de que disponemos, combinada con rutinas que manejen los puntos de la pantalla. Estas rutinas colocan los puntos al borde de la pantalla o los doblan de forma que aparecen en el otro lado de la pantalla. Veamos, pues, otros métodos para el dibujo de círculos con el ordenador, estudiando cómo pueden utilizarse con el Spectrum.

Método de la ecuación cuadrática

La primera técnica de dibujo de círculos construye un círculo

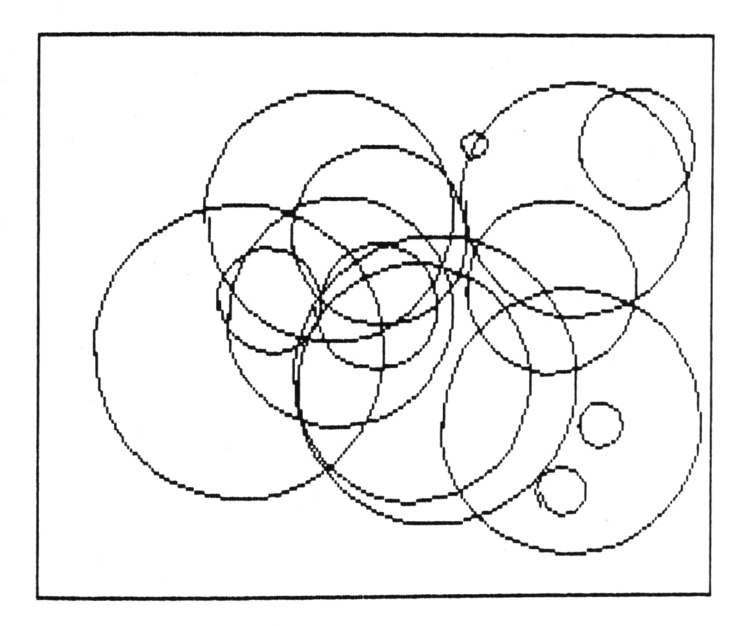


Fig. 4.4. Visualización típica de círculos aleatorios.

trazando gran número de puntos en posiciones calculadas utilizando la fórmula matemática de un círculo.

Comencemos por un pequeño segmento de círculo como el que se muestra en la figura 4.5. En este caso, el punto A está en el centro del círculo, mientras que los puntos B y C están sobre el mismo círculo. Las líneas AB y AC tendrán cada una de ellas una longitud igual al radio R del círculo.

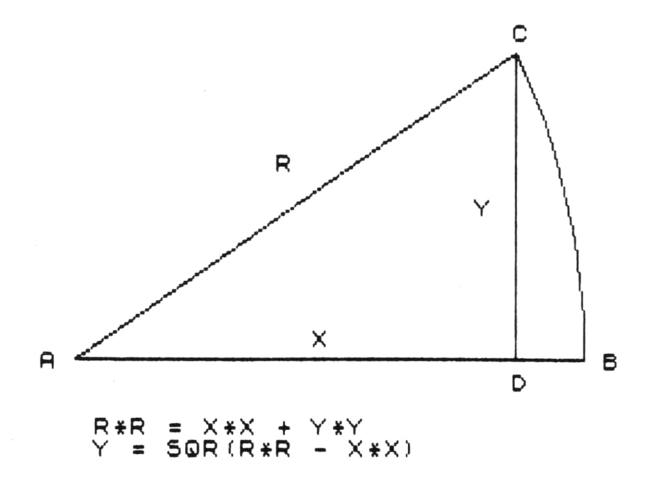


Fig. 4.5. Diagrama que muestra la utilización del método cuadrático para el dibujo de círculos.

En este diagrama, la línea AB es horizontal. Dibujemos ahora una línea vertical hacia abajo, desde el punto C hasta alcanzar el lado AB en D. Con ello, tenemos un triángulo rectángulo, ACD. Para dibujar un punto en cada uno de los puntos B y C, necesitaremos saber las coordenadas X e Y de cada uno de esos puntos, y es conveniente calcularlas con respecto al punto A que es el centro del círculo. El valor de X para el punto B viene dado por la longitud del lado AB, que es igual al radio R, y el valor Y será ∅, ya que el punto B se encuentra en la horizontal del punto A. Si miramos ahora el punto C, su valor de X será la longitud del lado AD, y su valor Y tendrá la misma longitud que el lado CD del triángulo ACD.

En un triángulo rectángulo, el cuadrado de la longitud del lado más largo es igual a la suma de los cuadrados de los otros dos lados, famoso Teorema de nuestro amigo Pitágoras.

Utilicemos el teorema para calcular las X e Y del punto C. Si aplicamos la regla al triángulo ACD, tenemos:

(AC)*(AC)=(AD)*(AD) + (CD)*(CD)

o bien con las variables que utilizamos para esos lados:

R * R = (X*X) + (Y*Y)

que, adecuadamente colocado, nos da:

Y * Y = (R*R) — (X*X)

ecuación de la que podemos sacar Y tomando la raíz cuadrada. Nuestro cálculo de la Y será por tanto:

Y = SQR ((R*R)—(X*X))

En esta ecuación, los valores de X e Y están medidos con referencia al punto centro del círculo. Observe que para el punto B, la ecuación sigue siendo válida, ya que este caso X = R, y el término de la derecha se hace ∅, por tanto Y = ∅.

Para colocar el círculo en un punto cualquiera de la pantalla, tendremos que añadir a las coordenadas X e Y, las del punto en el que se desea dibujar el círculo. Para evitar confusiones, llamaremos a estas coordenadas cx y cy.

Para trazar todos los puntos que forman el círculo, necesitamos calcular los valores de y para una serie de valores de x que van desde

—r hasta +r. Y cuantos más puntos calculemos, mejor será el dibujo del círculo.

Cada vez que se efectúa la raíz cuadrada de un número, existen dos soluciones posibles, para el mismo valor numérico, una de ellas es positiva y la otra negativa. Así pues, para cada valor de X se trazan dos puntos. Para trazar el primer punto del par, añadiremos el resultado de la raíz cuadrada al valor de Y en el centro del círculo, para obtener un punto situado por encima de la línea central del círculo. El segundo punto se halla restando la raíz cuadrada del valor Y del centro, y el punto quedará por debajo de la línea central del círculo. Como estamos trazando puntos individuales, y la semicircunferencia del círculo es unas tres veces su radio R, es conveniente tener un total de puntos igual a 4 veces R, para el trazado del círculo. Recuerde que dibujamos dos puntos para cada valor calculado, y, por tanto, un buen valor para el número de cálculos puede ser el doble del valor de radio R. Esto último es fácil de conseguir tomando todos los valores de X desde X = —R hasta X = +R. De este modo, un círculo con un radio de 50 unidades de pantalla necesitará 100 pasos, y dibujará un total de unos 200 puntos alrededor del círculo.

El programa que se muestra en la figura 4.6. dibuja círculos con centros en posiciones aleatorias en la pantalla. En este programa se hacen comprobaciones para los puntos que se salen de pantalla, corrigiéndolos para que permanezcan en el borde y eviten el fallo del programa, dando error por valores fuera de la gama.

```
100 REM Trazado de circulos por el
metodo cuadratico.
110 LET cx=128
120 LET cy=96
130 LET r=50
140 FOR x=-r TO r
150 LET y=SQR (r*r-x*x)
160 IF cx+x>255 THEN LET x=255-cx
170 IF cx+x<0 THEN LET x=0-cx
180 IF cy+y>175 THEN LET y=175-cy
190 IF cy+y<0 THEN LET y=0-cy
200 PLOT cx+x,cy+y
210 PLOT cx+x,cy-y
220 NEXT x
```

Fig. 4.6. Programa que dibuja círculos por el método cuadrático.

Con esta rutina, el número de cálculos depende del tamaño del círculo, y veremos que los círculos más grandes toman un tiempo considerable de elaboración. Esto se debe a que el ordenador tiene

que llevar a cabo muchos cálculos. Además la operación de extracción de la raíz cuadrada es una operación muy lenta en BASIC. Si deseamos dibujar círculos más rápidamente, necesitaremos utilizar otros medios de cálculo de los puntos que rodean el círculo.

Usted observará que, a los lados derecho e izquierdo del círculo, los primeros puntos tienden a espaciarse, especialmente en los círculos de gran radio. Podemos superar este inconveniente trazando más puntos, incrementando X por pasos de 0.5, por ejemplo, en lugar de pasos de 1.

Método trigonométrico

En lugar de dibujar series de puntos, podemos dibujar series de líneas cortas, que, al juntarse unas con otras, formen la circunferencia del círculo. Para el segundo método de dibujar círculos, necesitaremos utilizar algunos conocimientos sencillos de trigonometría.

La figura 4.7. muestra un segmento del círculo con dos puntos, B y C, sobre el círculo, y un punto A en el centro del círculo. Para dibujar el segmento del círculo, dibujaremos la línea BC.

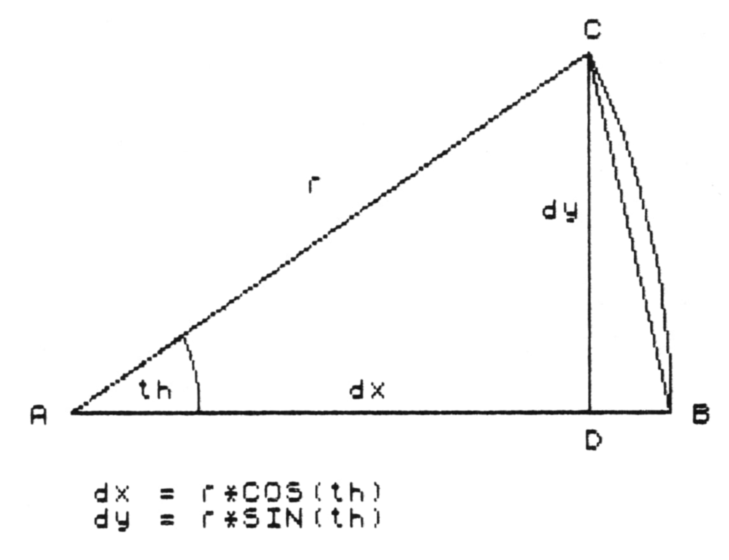


Fig. 4.7. Obtención del método trigonométrico para dibujar círculos.

Para dibujar el lado BC, necesitamos conocer las coordenadas del punto C. Tracemos una línea vertical desde C hasta D. La coordenada x del punto C viene dada por la longitud de la línea AD,

y la coordenada y vendrá dada por la longitud de la línea CD. En este punto es cuando entra la trigonometría.

Llamaremos THETA (la letra griega θ) al ángulo del punto A del triángulo (esta letra griega theta, se usa normalmente para representar ángulos). En nuestro programa utilizaremos, pues, la variable "th" para representar al ángulo theta.

Para hallar la longitud del lado CD, la función adecuada es SIN(theta)—(seno de theta). La definición de SIN(theta) es que es la relación entre la longitud del lado del triángulo opuesto al ángulo theta y la longitud de la hipotenusa (lado opuesto al ángulo recto), en nuestro caso, AC.

Por lo tanto, en el triángulo anterior:

SIN(th) = CD/AC

Por el momento sabemos que AC es igual al radio r.

La longitud del lado CD es el cambio que necesitamos realizar con y, para obtener el valor de y en el punto c. A este cambio le llamaremos dy. Sustituyendo estos términos en la ecuación anterior, tenemos:

SIN(th) = dy/r

multiplicando los dos miembros por r, resultará,

dy = r * SIN (th)

Una vez hallado el valor de dy, necesitamos encontrar el valor del lado AD, que será el cambio que debe sufrir la x, y que llamaremos dx. Como el COS(th) es la relación entre la longitud del lado adyacente (AD) del triángulo y la longitud de la hipotenusa (AC), tendremos:

COS(th) = AD/AC

sustituyendo los valores dx y r tendremos:

dx = r * COS(th)

Para encontrar las coordenadas del siguiente punto del círculo, aplicamos la misma ecuación, pero en este caso el ángulo tiene un valor diferente.

Para dibujar el círculo debemos comenzar por colocar el cursor en el punto B, para el cual dx=r y dy=Ø. Este paso se lleva a cabo fijando los valores de las variables x1 =cx +r, y y1 =cy. Utilizando ahora:

PLOT x1,y1

trazaremos el primer punto. Las variables cx y cy son las coordenadas del centro del círculo.

El siguiente paso es calcular las coordenadas del punto C que vienen dadas por las ecuaciones:

x2 = cx + dx = cx + r *SIN(th)
y2 = cy + dy = cy + r *COS(th)

y utilizando x2,y2, podemos dibujar la línea BC del modo siguiente:

DRAW x2—x1,y2—y1

Para el siguiente segmento de línea del círculo, los valores de x1 e y1 se hacen iguales a los valores de x2 e y2. El ángulo th se incrementa, y se calculan nuevos valores para x2,y2, utilizando el nuevo valor del ángulo th. Este proceso continúa hasta que el ángulo alcanza 36Ø grados y se ha dibujado un círculo completo.

¿Cuál es el criterio para decidir el valor de th? Una rotación completa del ángulo th alrededor del círculo es de 360 grados. Para realizar un círculo uniforme, cuantos más pasos demos mejor.

Un valor práctico para el número de pasos es el de las unidades del radio r. Suponga que desea un círculo de radio 6Ø. En este caso, cada segmento del círculo añade 36Ø/6Ø, es decir, 6 grados al valor de th.

Aunque estamos familiarizados con los grados, el computador no trabaja con ellos, trabaja con radianes. Sólo necesitamos conocer que 36Ø es igual a 2*PI radianes, por tanto, el ángulo se incrementa en 2*PI/60 radianes por cada segmento de círculo.

```
100 REM Dibujo de circulos
110 REM Utilizando el sistema trigonometrico
120 BORDER 6
130 FOR s=1 TO 10
140 CLS
150 FOR n=1 TO 10
160 LET r=10+(RND*40)
170 LET x=INT (RND*255)
180 LET y=INT (RND*175)
```

```
190 GO SUB 500
200 NEXT n
210 PAUSE 100
220 NEXT s
230 STOP
500 REM Subrutina de circulos.
510 LET dt=2*PI/r
520 REM Establece el punto de partida
530 LET th=0
540 LET x1=x+INT (r*COS th)
550 LET y1=y+INT (r*SIN th)
560 REM Corrige los puntos que se salen
de la pantalla.
570 IF x1>255 THEN LET x1=255
580 IF x1<0 THEN LET x1=0
590 IF y1>175 THEN LET y1=175
600 IF y1<0 THEN LET y1=0
610 PLOT x1,y1
620 FOR i=0 TO r
630 LET th=th+dt
640 LET x2=x+INT (r*COS th)
650 LET y2=y+INT (r*SIN th)
655 REM Corrige los puntos que se salen
de la pantalla.
660 IF x2>255 THEN LET x2=255
670 IF x2<0 THEN LET x2=0
680 IF y2>175 THEN LET y2=175
690 IF y2<0 THEN LET y2=0
700 DRAW x2-x1,y2-y1
710 LET x1=x2
720 LET y1=y2
730 NEXT i
740 RETURN
```

Fig. 4.8. Círculos aleatorios utilizando el método trigonométrico.

El número PI es una constante cuyo valor es aproximadamente 3.14, y es la relación entre la circunferencia de un círculo y su diámetro. Nosotros no necesitamos recordar el valor de PI, porque el Spectrum tiene una tecla especial que nos permite introducir PI en una instrucción, como constante. Para introducir PI en una instrucción, pulse al mismo tiempo CAPS SHIFT y SIMBOL SHIFT para obtener el modo extendido (E). Luego pulse la tecla M.

El dibujo de un círculo supone la utilización de un sencillo bucle para repetir los cálculos y dibujar un pequeño segmento de línea r veces. Después de dibujar cada segmento de círculo, el valor de th se incrementa, y los valores de x1 e y1 se actualizan con los del final de la línea que se acaba de trazar, preparándolos para el paso siguiente.

Para dibujar nuestro círculo, calculamos únicamente una serie de valores de x e y para unos valores de theta desde Ø a 36Ø grados (Ø a 2*PI radianes). El número de puntos que necesitemos depende del

tamaño del círculo y de la precisión con que queramos dibujarlo. Para nosotros, una buena figura es la que tiene un número de unidades igual al radio, es decir, para un círculo de 50 de radio podemos utilizar unos 50 puntos. El ángulo theta para cada paso se halla dividiendo simplemente 2*PI por el número de puntos que deseamos, por tanto, el ángulo por paso será 2*PI/R. En la figura 4.8. se muestra un programa para el dibujo de círculos que utiliza esta técnica. Este programa dibuja una serie de círculos aleatoriamente por toda la pantalla, y da un resultado similar al que se muestra en la figura 4.9.

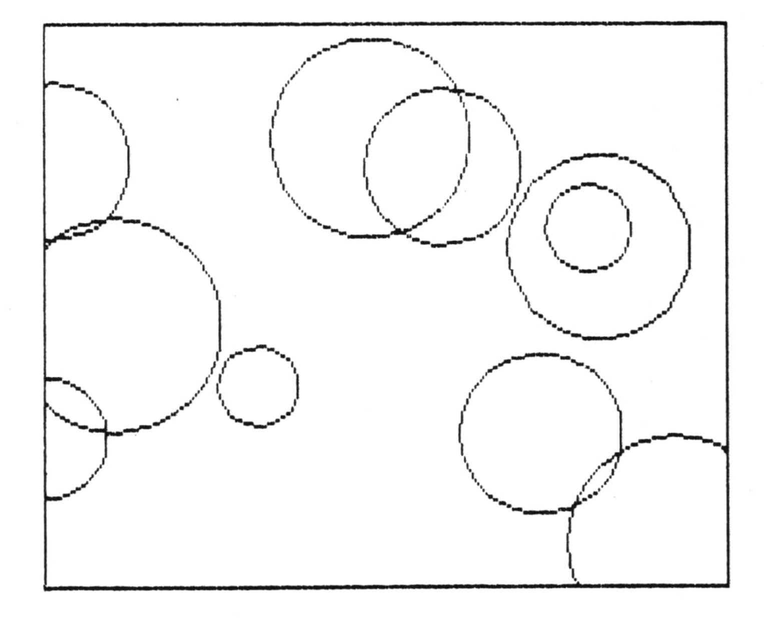


Fig. 4.9. Dibujo típico correspondiente al programa 4.8.

El programa de dibujo del círculo se ha escrito como subrutina, comenzando en la línea 5ØØ. Si se dibujan círculos con frecuencia, es conveniente utilizar una subrutina para el dibujo de círculos, y llamarla desde el programa principal cada vez que se necesite dibujar un círculo. Algunas veces, los valores calculados para las coordenadas x e y pueden caer fuera de los límites de la pantalla, y, si se utilizan en una instrucción DRAW, producirán un mensaje de error, parando el programa. Para evitar este problema se comprueban x2 e y2, y si se salen de los límites establecidos, se fijan en el valor límite de la pantalla. Por tanto, si x2<Ø, x2 se fija en Ø. Esto produce una

línea al borde de la pantalla, y parte del círculo estará fuera de este límite de pantalla.

Método de rotación

Otro método diferente para el cálculo de los valores x,y de un círculo, es basarse en el ángulo que rota cada vez la línea radial. En este caso, los nuevos valores para x2 e y2 se calculan a partir de los valores del punto anterior (x1,y1) y no del radio y ángulo total.

Si miramos la figura 4.10., el valor de y1 es cero, y, por lo tanto, el resultado sólo se ve afectado por el valor de x1 que tiene un valor igual a r. Así pues:

x2 = x1 *COS (th)
y2 = x1 *SEN (th)

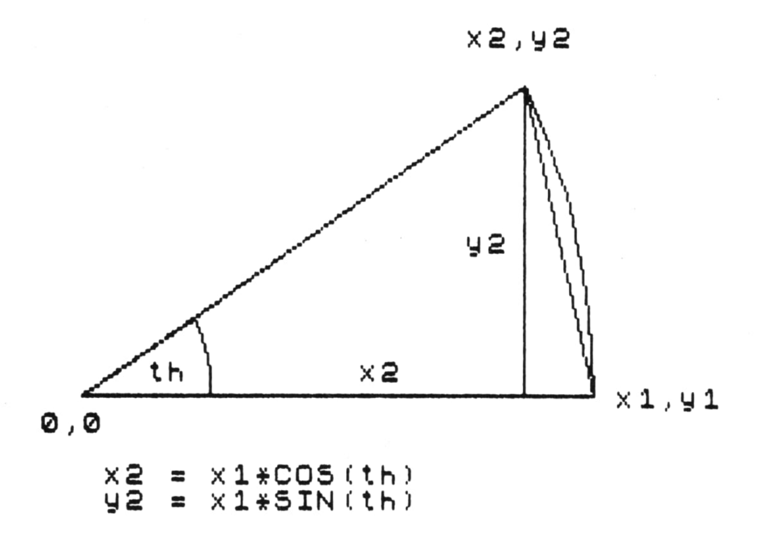


Fig. 4.10. Rotación del eje x.

Consideremos ahora la situación en la que la línea radial es vertical y se va moviendo un ángulo theta. En la figura 4.11. podemos ver que el valor de x1 es 0, y el resultado sólo se ve afectado por el valor de y1. En este caso, el valor de x2 es negativo, puesto que se ha movido a la izquierda de la línea donde x1 =0. Los resultados son:

x2 = —y1 *SIN(th)
y2 = y1*COS(th)

Si combinamos estos dos resultados, podemos producir la expresión general para el cálculo de x2 e y2 con valores iniciales cualesquiera x1, y1.

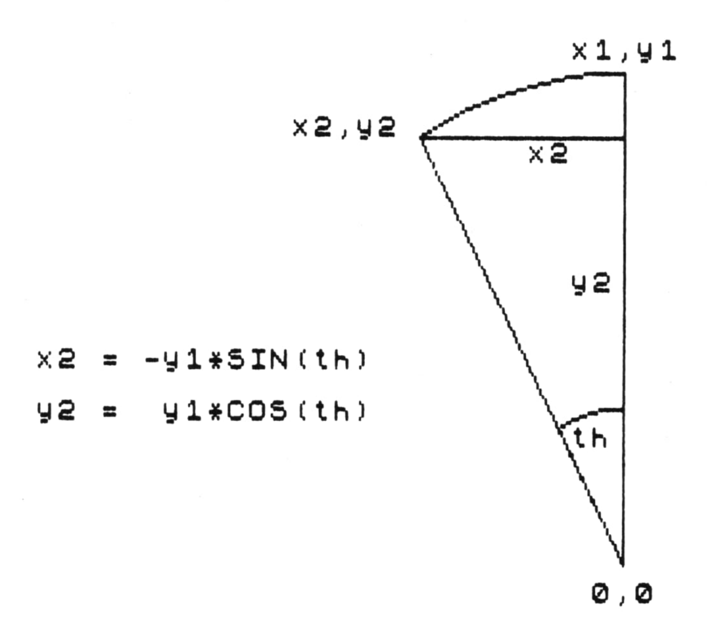


Fig. 4.11. Rotación del eje y.

Las dos nuevas ecuaciones son:

x2 = x1 *COS(th) — y1 *SIN(th)
y2 = x1 *SIN(th) — y1 *COS(th)

La mayor ventaja de este método es que el valor de th es constante y podemos calcular los valores de SIN(th) y COS(th) antes de empezar con el cálculo de coordenadas y el bucle de dibujo de las líneas, eliminando virtualmente todos los cálculos trigonométricos, que suelen ser muy lentos. El programa para dibujar un círculo tiene ahora la forma del listado de la figura 4.12.

```
100 REM Trazado de circulos aleatorios
110 REM por el metodo de rotacion
120 FOR n=1 TO 15
130 LET r=10+INT (RND*40)
```

```
140 LET x=r+INT (RND*(255-2*r))
150 LET y=r+INT (RND*(155-2*r))
160 GO SUB 500
170 NEXT n
180 STOP
500 REM Subrutina del circulo.
510 LET th=2*PI/r
520 LET x1=r
530 LET y1=0
540 PLOT x+x1,y+y1
550 FOR i=1 TO r
560 LET x2=x1*COS th-y1*SIN th
570 LET y2=x1*SIN th+y1*COS th
580 DRAW x2-x1,y2-y1
590 LET x1=x2
600 LET y1=y2
610 NEXT i
620 RETURN
```

Fig. 4.12. Dibujo de un círculo por el método de rotación.

Dibujo de polígonos

Si reducimos el número de pasos, y, por tanto, el número de segmentos de línea utilizados en la rutina de dibujo de círculos, el resultado será una figura con muchos lados iguales. Este tipo de figura se llama *POLIGONO* regular. Si utilizamos el método trigonométrico, los pasos para cada ángulo th resultan muy grandes. Supongamos que dibujamos un polígono de ocho lados, llamado *octógono*, en este caso, el ángulo cambiará 2*PI/8 por cada paso en el dibujo. El programa listado en la figura 4.13. dibujará un *octógono* en el centro de la pantalla. Si lo que deseamos es una figura de seis lados, un *HEXAGONO*, el número de pasos en el bucle de dibujo se reduce a 6. Así pues, el ángulo total de 2*PI se divide por 6 para dar un cambio de th de 2*PI/6 por cada paso.

Para realizar una rutina general para el trazado de polígonos, podemos introducir un nuevo parámetro ns(número de lados) y modificar el programa para que realice el número de pasos de trazado que deseamos. El cambio de th para cada paso (dt) vendrá dado por:

dt = 2*PI/ns

```
100 REM Programa que dibuja un  octogono.
110 LET xc=128
120 LET yc=88
130 LET r=50
140 LET dt=2*PI/8
150 LET th=0
160 LET x1=xc+r
170 LET y1=yc
180 PLOT x1,y1
190 FOR n=1 TO 8
200 LET th=dt*n
210 LET x2=xc+r*COS th
220 LET y2=yc+r*SIN th
230 DRAW x2-x1,y2-y1
240 LET x1=x2
250 LET y1=y2
260 NEXT n
```

Fig. 4.13. Programa para dibujar un octógono.

Para ver cómo funciona, haga el programa que se muestra en la figura 4.14. Este programa dibuja una serie de figuras, cuyo tamaño va aumentando, centradas en un punto próximo al centro de la pantalla (véase la figura 4.15). Las figuras van incrementando su número de lados, comenzando por un triángulo.

```
100 REM Poligonos inscritos unos en otros
(anidados)
110 LET r=12
120 LET xc=128
130 LET yc=88
140 FOR k=3 TO 9
145 REM Establece el numero de lados
150 LET ns=k
160 LET dt=2*PI/ns
170 LET x1=xc+r
180 LET y1=yc
190 PLOT x1,y1
195 REM Dibuja el poligono.
200 FOR n=1 TO ns
210 LET th=n*dt
220 LET x2=xc+r*COS th
230 LET y2=yc+r*SIN th
240 DRAW x2-x1,y2-y1
250 LET x1=x2
260 LET y1=y2
270 NEXT n
280 LET r=r+12
290 NEXT k
```

Fig. 4.14. Visualización de pantalla producida por la figura 4.14.

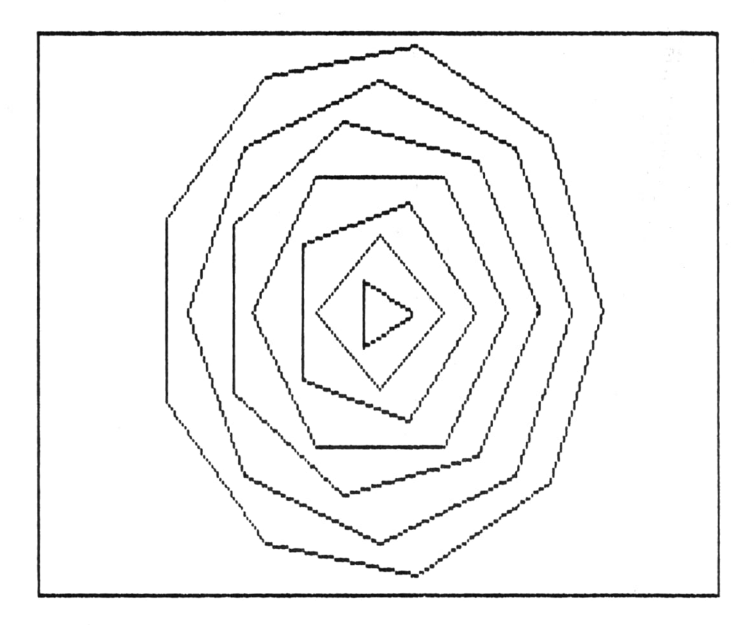


Fig. 4.15. Visualización de pantalla producida por la figura 4.14.

Dibujo de polígonos por rotación

Ahora que podemos dibujar círculos utilizando la técnica de la línea de rotación, también podemos dibujar octógonos y otros polígonos. Para un octógono tendremos un ángulo th de 2*PI/8, y, por tanto, una subrutina que dibuje un octógono puede ser la que se muestra en la figura 4.16.

Como antes, podemos desarrollar esta rutina y hallar una rutina general de dibujo de polígonos añadiendo la variable k (número de lados). Podremos, pues, producir un programa que dibuje polígonos desde 3 a 12 lados, con diversos tamaños, por toda la pantalla. (Ver figura 4.17).

Este procedimiento de trazado de polígonos puede usarse como método general para cualquier programa. Observe que dibujará cuadrados y triángulos, pero los triángulos serán siempre equiláteros.

```
100 REM Octogono realizado por el metodo de
rotacion.
110 LET xc=128
120 LET yc=88
130 LET r=50
140 GO SUB 500
```

```
150 STOP
500 REM Subrutina de dibujo del octogono.
510 LET x1=r
520 LET y1=0
530 LET th=2*PI/8
540 LET sn=SIN th
550 LET cn=COS th
560 PLOT xc+x1,yc+y1
570 FOR n=1 TO 8
580 LET x2=x1*cn-y1*sn
590 LET y2=x1*sn+y1*cn
600 DRAW x2-x1,y2-y1
610 LET x1=x2
620 LET y1=y2
630 NEXT n
640 RETURN
```

Fig. 4.16. Dibujo de un octógono por el método de rotación.

```
100 REM Poligonos aleatorios.
110 FOR j=1 TO 20
120 LET r=10+INT (RND*40)
130 LET xc=r+INT (RND*(255-2*r))
140 LET yc=r+INT (RND*(175-2*r))
150 LET k=3+INT (RND*5)
160 GO SUB 500
170 NEXT j
180 STOP
490 REM Subrutina del poligono.
500 LET th=2*PI/k
510 LET sn=SIN th
520 LET cn=COS th
530 LET dx=r
540 LET dy=0
550 PLOT xc+dx,yc+dy
560 FOR n=1 TO k
570 LET xr=dx*cn-dy*sn
580 LET yr=dx*sn+dy*cn
590 DRAW xr-dx,yr-dy
600 LET dx=xr
610 LET dy=yr
620 NEXT n
630 RETURN
```

Fig. 4.17. Programa para realizar polígonos aleatoriamente.

Figuras con forma de estrella y coronas circulares

La rutina de dibujo de polígonos puede modificarse fácilmente para dibujar figuras con forma de estrella. En este caso, se dibuja una línea desde el centro de cada punto calculado (como para el caso del polígono), cambiando el procedimiento de dibujo. El programa listado en la figura 4.18. dibuja aleatoriamente un esquema de estrella sobre la pantalla.

En este caso, la línea se dibuja desde cada vértice del polígono hacia el centro, es decir, las líneas salen radiadas desde el centro, como los radios de una rueda.

Las coronas circulares (formas de rueda) pueden dibujarse trazando primero la estrella, y luego dibujando un círculo del mismo radio con el mismo punto central.

```
100 REM Figuras en forma de estrella.
110 FOR j=1 TO 20
120 LET r=10+INT (RND*40)
130 LET xc=r+INT (RND*(255-2*r))
140 LET yc=r+INT (RND*(175-2*r))
150 LET k=3+INT (RND*5)
160 GO SUB 500
170 NEXT j
180 STOP
490 REM Subrutina que forma las estrellas.
500 LET th=2*PI/k
510 LET sn=SIN th
520 LET cn=COS th
530 LET dx=r
540 LET dy=0
550 FOR n=1 TO k
560 LET xr=dx*cn-dy*sn
570 LET yr=dx*sn+dy*cn
580 PLOT xc,yc
590 DRAW xr,yr
600 LET dx=xr
610 LET dy=yr
620 NEXT n
630 RETURN
```

Fig. 4.18. Programa que dibuja figuras con forma de estrella.

Escala y alargamiento

En el dibujo de cuadrados, polígonos y círculos, el tamaño de una figura determinada dependía del valor de W ó R que utilizáramos en la rutina. Así pues, alterando W ó R podemos alterar el tamaño o escala de la figura.

En el caso del rectángulo existen dos datos que influyen en la escala, uno es la anchura (W) y otro la altura (H). En efecto, así vemos que el cuadrado es un rectángulo que ha sido aplastado o comprimido en una dirección. Suponiendo que lo comprimimos sólo horizontalmente o verticalmente, veremos que esto significa únicamente que los valores de la escala para x e y son diferentes.

Podemos aplicar esta idea de estiramiento a otras figuras utilizando dos variables más, que serán los factores x e y de la escala. Para llegar a un resultado correcto, el punto de referencia alrededor

del cual se va a trazar la figura, debe estar en el centro de ésta. Esto es siempre verdad con polígonos y circuitos y con los métodos que hemos utilizado para dibujarlos. En estos casos, los factores de escala se utilizan como multiplicadores de los términos dx y dy en los cálculos del dibujo. Observe que los factores de escala no se aplican a las coordenadas cx y cy alrededor de las cuales se traza la figura.

Consideremos un círculo, y utilicemos el método de trazado trigonométrico como el que se muestra en la figura 4.19. Ahora se utilizarán dos nuevos términos sx y sy, y los valores de dx y dy se multiplican por sx y sy, respectivamente, antes del trazado de la figura. Comencemos dibujando el círculo a la izquierda de la pantalla, incrementando sy en pasos desde Ø hasta 1, y con sx constante e igual a 1. A continuación, dibujaremos el círculo a la derecha de la pantalla con sx, aumentando desde Ø hasta 1 y sy con valor fijo de 1. Finalmente, variaremos ambos valores, sx y sy, desde Ø hasta 1.

Si sx y sy valen 1 los dos, la figura obtenida será un círculo. Si sx<1 y sy⩽1, la figura pasa a ser una elipse cuyo eje mayor es el horizontal. Si sx>1, y sy⩾1, la elipse tendrá su eje mayor vertical. Si sx o sy son negativos, simplemente el efecto será como si lo hubiéramos pintado al revés. Si la figura que teníamos no era simétrica, el lado izquierdo estará a la derecha, o bien la parte superior estará abajo, dando el efecto de la imagen de un espejo.

```
100 REM Factores de escala aplicados al circulo.
120 LET xc=40
130 LET yc=88
140 LET r=40
145 REM Escala aplicada a la y.
150 FOR a=1 TO 0 STEP -.2
160 LET sx=1
170 LET sy=a
180 GO SUB 500
190 NEXT a
200 LET xc=210
205 REM Escala aplicada a la x.
210 FOR a=1 TO 0 STEP -.2
220 LET sx=a
230 LET sy=1
240 GO SUB 500
250 NEXT a
260 LET xc=128
265 REM Escalas aplicadas a la y,y luego a la x.
270 FOR a=1 TO 0 STEP -.2
280 LET sx=1
290 LET sy=a
300 GO SUB 500
310 NEXT a
320 FOR a=1 TO 0 STEP -.2
330 LET sx=a
```

```
340 LET sy=1
350 GO SUB 500
360 NEXT a
370 STOP
490 REM Subrutina de circulos.
500 LET dt=2*PI/r
510 LET x1=xc+sx*r
520 LET y1=yc
530 PLOT x1,y1
540 FOR n=1 TO r
550 LET x2=xc+sx*r*COS (n*dt)
560 LET y2=yc+sy*r*SIN (n*dt)
570 DRAW x2-x1,y2-y1
580 LET x1=x2
590 LET y1=y2
600 NEXT n
610 RETURN
```

Fig. 4.19. Demostración de cómo se dibuja una elipse.

Rotación de figuras

Las figuras que hemos dibujado hasta ahora tenían todas su eje x horizontal. Suponga, sin embargo, que deseamos dibujar un rectángulo, pero queremos visualizarlo tumbado, con un ángulo como el que se muestra en la figura 4.20.

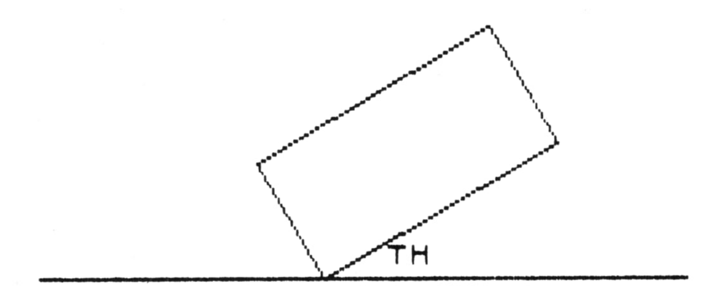


Fig. 4.20. Diagrama que muestra una figura rotada un ángulo TH.

Hemos visto ya que un punto puede rotar con respecto a otro punto de la pantalla, y que esta técnica se utilizaba para dibujar polígonos y círculos. Si podemos rotar un punto, también podremos rotar fácilmente todos los puntos de la figura. En este caso, las ecuaciones de rotación se aplican a cada punto de la figura. Cuando la figura se dibuja utilizando este nuevo conjunto de puntos, estará inclinada con respecto a la horizontal. Para encontrar los valores de la rotación para DX y DY, utilizamos:

XR = DX*COS(TH) — DY*SIN(TH)

y

YR = DX*SIN(TH) + DY*COS(TH)

donde DX y DY son las coordenadas de cada punto, medidas con relación al punto sobre el que queremos rotar la figura. El ángulo de rotación es de TH radianes con el eje positivo de las x.

Comencemos con un rectángulo y supongamos que estamos utilizando como referencia el vértice inferior izquierdo, punto sobre el que rotaremos el rectángulo. Supondremos que el rectángulo tiene una anchura W y una altura H, y que el ángulo que queremos girarlo es TH.

Podemos comenzar fijando x1 e y1 como las coordenadas del punto de referencia del vértice inferior izquierdo, cuyas coordenadas actuales de pantalla son xc e yc. La primera línea que debe dibujarse es la del fondo, y, por tanto, el primer punto que debe girar es el del vértice inferior derecho. Para este punto, la X y la Y equivalen a DX y DY, medidas desde el vértice inferior izquierdo, cuyas coordenadas actuales de pantalla son XC,YC. DX = W y DY = Ø. Ahora, ya podemos calcular la posición después del giro del punto (XR,YR), introduciendo los valores de DX y DY en las ecuaciones de rotación.

Para dibujar la primera línea, empezamos dando los valores X1 e Y1 a XC e YC, respectivamente, y trazaremos un punto. Después, calcularemos las coordenadas del otro extremo de la línea, X2,Y2 añadiendo los valores XR e YR a las coordenadas de referencia XC,YC, como puede verse,

X2 = XC + YR
Y2 = YC+YR

La primera línea puede dibujarse utilizando las coordenadas X1,Y1 y X2,Y2, de los dos finales de la línea, en la siguiente instrucción DRAW.

DRAW X2—X1,Y2—Y1

Para dibujar el segundo lado del rectángulo, hacemos X1,Y1 igual a los valores X2,Y2, para que la nueva línea comience desde ese punto, final de la primera línea. El valor DX para el segundo punto seguirá siendo igual a W, pero el valor DY es ahora igual a la altura

H. Con estos nuevos valores podemos aplicar las ecuaciones de rotación y obtener nuevos valores para XR e YR, y para X2 e Y2. Ya podemos dibujar el segundo lado. Este proceso se repite entonces para los otros dos lados que nos quedan. Como la rotación y los pasos del dibujo se repiten, es conveniente hacerlos en forma de subrutina. El programa de dibujo de un rectángulo girado, será, pues, el que se muestra en la figura 4.21.

```
100 REM Rectangulo al que se aplica una rotacion.
110 LET xc=128
120 LET yc=40
130 LET w=100
140 LET h=30
150 LET dt=PI/6
160 LET th=0
170 FOR n=1 TO 6
180 LET x1=xc
190 LET y1=yc
200 PLOT x1,y1
210 LET x2=xc+w*COS th
220 LET y2=yc+w*SIN th
230 DRAW x2-x1,y2-y1
240 LET x1=x2
250 LET y1=y2
260 LET x2=xc+w*COS th-h*SIN th
270 LET y2=yc+w*SIN th+h*COS th
280 DRAW x2-x1,y2-y1
290 LET x1=x2
300 LET y1=y2
310 LET x2=xc-h*SIN th
320 LET y2=yc+h*COS th
330 DRAW x2-x1,y2-y1
340 LET x1=x2
350 LET y1=y2
360 LET x2=xc
370 LET y2=yc
380 DRAW x2-x1,y2-y1
390 LET th=th+dt
400 NEXT n
```

Fig. 4.21. Programa para dibujar una serie de rectángulos girados.

El efecto de rotación combinado con cambios de escala puede producir diseños de espirales interesantes. En la figura 4.22 se muestra un programa de este tipo.

Esta técnica de rotación puede aplicarse a cualquiera de las figuras que se generan por una serie de pasos matemáticos. Sin embargo, si tenemos una figura irregular, las cosas son más difíciles. Para este tipo de figuras es mejor realizar una tabla con los datos de los puntos de los diferentes vértices. Estos puntos X e Y se miden con relación a un punto de la figura alrededor del cual se produce el giro. Este

```
100 REM Patrones trazados
110 REM utilizando rotaciones y escalas.
120 LET th=0
130 LET dt=PI/12
140 LET xc=128
150 LET yc=88
160 FOR w=2 TO 70 STEP 2
170 LET x1=xc
180 LET y1=yc
185 PLOT x1,y1
190 LET dx=w
200 LET dy=0
210 GO SUB 500
220 LET x1=x2
230 LET y1=y2
240 LET dx=w
250 LET dy=w
260 GO SUB 500
270 LET x1=x2
280 LET y1=y2
290 LET dx=0
300 LET dy=w
310 GO SUB 500
320 LET x1=x2
330 LET y1=y2
340 LET dx=0
350 LET dy=0
360 GO SUB 500
370 LET th=th+dt
380 NEXT w
390 STOP
490 REM Subrutina de rotacion.
500 LET x2=xc+dx*COS th-dy*SIN th
510 LET y2=yc+dx*SIN th+dy*COS th
520 DRAW x2-x1,y2-y1
530 RETURN
```

Fig. 4.22. Diseños utilizando escalas y rotaciones.

punto puede ser el centro de la figura, o bien puede encontrarse fuera de ella. Para dibujar la figura, sencillamente tomamos los puntos uno detrás del otro, y dibujamos las líneas que los unen. Sin embargo, surge un problema. En algunos casos queremos mover un punto sin dibujar la línea. Este problema puede arreglarse produciendo una tercera matriz de datos, que llamaremos L. Si L vale 1, se dibuja la línea, y si L vale Ø, no se dibuja.

Una vez creada la tabla de los valores X,Y y L, podemos proceder al dibujo de la figura, utilizando la misma técnica que antes para dibujar el rectángulo, con la diferencia que ahora los valores X e Y se van tomando sucesivamente de la matriz.

Los valores para X1,Y1 se fijan inicialmente en las coordenadas de pantalla alrededor de las cuales vamos a dibujar la forma, y entonces X2,Y2 se calculan utilizando las ecuaciones de rotación, y

añadiendo los valores ya rotados a XC e YC, respectivamente. Antes de dibujar la línea, se comprueba L, y, si es Ø, se salta el comando DRAW. Si no se dibuja la línea, el comando PLOT para el comienzo de la nueva línea mueve el cursor gráfico a esa posición, preparado para dibujar la nueva línea. Después de procesar cada línea, los valores de X1 e Y1 se actualizan con los valores de X2,Y2, y se preparan para el proceso de la siguiente línea. Este proceso continúa hasta completar la figura.

Si hay varios puntos en la figura, podemos acelerar algo la rutina. Como el ángulo TH es constante para todos los puntos de la figura, los valores de SIN(TH) y COS(TH) pueden calcularse antes de comenzar el bucle de dibujo. Ahora podemos utilizar los resultados SN y CN en el interior del bucle, evitando muchos cálculos

```
100 REM Rotacion de una figura irregular.
110 DIM x(5)
120 DIM y(5)
130 DIM l(5)
135 REM Establecimiento de los  datos de la
figura.
140 FOR n=1 TO 5
150 READ x(n),y(n),l(n)
160 NEXT n
170 DATA 20,0,0
180 DATA 60,0,1
190 DATA 60,-15,0
200 DATA 40,0,1
210 DATA 60,15,1
220 LET xc=128
230 LET yc=88
240 LET dt=2*PI/10
250 LET th=0
260 FOR f=1 TO 10
270 LET sn=SIN th
280 LET cn=COS th
290 LET x1=xc
300 LET y1=yc
310 GO SUB 500
320 LET th=th+dt
330 NEXT f
340 STOP
490 REM Subrutina de dibujo de la figura.
500 PLOT x1,y1
510 FOR n=1 TO 5
520 LET x2=xc+x(n)*cn-y(n)*sn
530 LET y2=yc+x(n)*sn+y(n)*cn
540 IF l(n)=1 THEN DRAW x2-x1,y2-y1
550 PLOT x2,y2
560 LET x1=x2
570 LET y1=y2
580 NEXT n
590 RETURN
```

Fig. 4.23. Rotación de una forma irregular.

trigonométricos que hacen mucho más lenta la ejecución. El programa se convertirá, pues, en el que se muestra en la figura 4.23.

Aquí hemos usado las matrices X e Y para definir los puntos de la figura. Las variables XC e YC se han utilizado para definir el origen alrededor del cual se traza la figura sobre la pantalla.

Capítulo 5
Nuevos caracteres y formas

Hasta ahora hemos utilizado el conjunto normal de caracteres y los símbolos gráficos de mosaico para dibujar sobre la pantalla y producir las visualizaciones. Para muchas aplicaciones, esto resultará adecuado, pero habrá situaciones donde deseemos imprimir un símbolo que no existe en el conjunto standard de símbolos. Por ejemplo, podemos querer producir símbolos que representen los palos de la baraja, o quizás necesitemos símbolos del alfabeto griego para un programa matemático.

Una posible solución al problema puede ser dibujar en ese momento el símbolo que deseemos, utilizando los comandos de alta resolución PLOT y DRAW. Esto requiere dibujar el símbolo en un pedazo de papel y luego trabajar con la secuencia de pasos de trazado necesaria para producir la forma deseada sobre la pantalla. Si sólo necesitamos un símbolo o diseño, y la forma no es difícil de dibujar, esta aproximación puede ser razonablemente práctica, aunque tediosa. Si necesitamos varios símbolos diferentes, y, en especial, cuando los símbolos deban mezclarse con texto normal impreso, como en el caso de las letras griegas o rusas, el método de dibujo no resulta nada práctico. Lo que necesitamos realmente es un método de producción de símbolos especiales que puedan imprimirse del mismo modo que otros símbolos de texto.

El Spectrum, de hecho, nos permite que produzcamos un conjunto de símbolos diseñados a medida, y que podamos utilizarlos del mismo modo que si fueran caracteres standard. Veamos ahora cómo funciona esto.

Caracteres definidos por el usuario

Si imprimimos con el Spectrum todos los caracteres disponibles, utilizando el programa que se muestra al comienzo del Capítulo Dos, encontraremos que después del conjunto de símbolos de

mosaico se encuentran letras desde la A a la U. Puede parecer raro que se repitan los diseños de esos veinte símbolos. De hecho, éstos son los símbolos definidos por el usuario, los cuales están formados por un patrón de puntos para crear el símbolo, fijado por la persona que opera el equipo.

Reprogramando los patronatos de puntos para estos símbolos, podemos generar letras griegas o rusas, o incluso japonesas, de Katakana o caracteres Chinos. Además de los símbolos de texto, podemos programar los patrones de puntos de esos símbolos para visualizar invasores del espacio, piedras, palos de baraja, etc.

A diferencia del texto normal y de los símbolos de mosaico, que tienen sus patrones almacenados en la memoria ROM (Read Only Memory)-(Memoria de sólo lectura) los símbolos definidos por el usuario tienen sus patrones de puntos guardados en la memoria normal lectura-escritura. Cuando se conecta el Spectrum, éste copia automáticamente los patrones de puntos de las letras A a U, en una zona de la memoria reservada para los patrones de puntos de los símbolos definidos por el usuario. Si no se hiciera de este modo, los símbolos serían patrones aleatorios de puntos. La zona de memoria utilizada para los patrones a medida, se encuentra en la parte superior de la memoria principal. Las direcciones concretas de memoria dependerán de si el Spectrum es de 16K o de 48K.

Como los demás caracteres del texto, cada uno de los símbolos definidos por el usuario tiene 8 filas, con 8 puntos en cada fila, y cada punto puede activarse o no. En el ordenador, cada palabra de memoria tiene 8 bits, cada uno de los cuales puede activarse (1) o desactivarse (0), por tanto, es conveniente almacenar una fila de puntos de un patrón de carácter en una palabra de memoria. Las ocho filas de puntos que forman el carácter, se almacenan en ocho palabras de memoria sucesivas. Si queremos crear un nuevo símbolo, el nuevo patrón de puntos se escribirá en un conjunto de ocho posiciones de memoria, en el área de memoria reservada para los símbolos definidos por el usuario.

Programación de un nuevo símbolo

El primer paso para crear un nuevo símbolo es trabajar en el patrón de puntos que necesitamos para formarlo. Esto puede hacerse simplemente dibujando una cuadrícula con ocho filas de cuadrados y ocho cuadrados en cada fila, como se muestra en la Figura 5.1. Los cuadrados se van oscureciendo para obtener la forma del símbolo.

Una vez se ha trabajado el patrón de puntos, el siguiente paso es calcular los números que deben almacenarse en la memoria. La figura 5.1. muestra la visualización de un carácter típico diseñado por el usuario. De hecho es el mismo patrón básico de un monigote que ya realizamos utilizando los símbolos de mosaico. En el diagrama, los puntos negros son de color INK(texto), y se representarán por el "1" en la palabra de ordenador. Cada bit de datos en la palabra tiene un valor numérico que comienza en 1 para el bit del extremo derecho y sigue la secuencia 2,4,8,16, y así sucesivamente con los demás bits, según nos vamos moviendo hacia la izquierda en la palabra de datos. El valor que toma cada bit se muestra en la parte superior del diagrama.

Si un punto está en color INK, el bit de datos correspondiente en la palabra está en 1, pero si el punto está en el color PAPER (de fondo), el bit será un 0. Para encontrar el número decimal que en cada momento debe introducirse en el ordenador, simplemente sumaremos todos los valores numéricos para todos los bits en la palabra que tienen "1". Esto nos dará un número en el intervalo de 0 a 255.

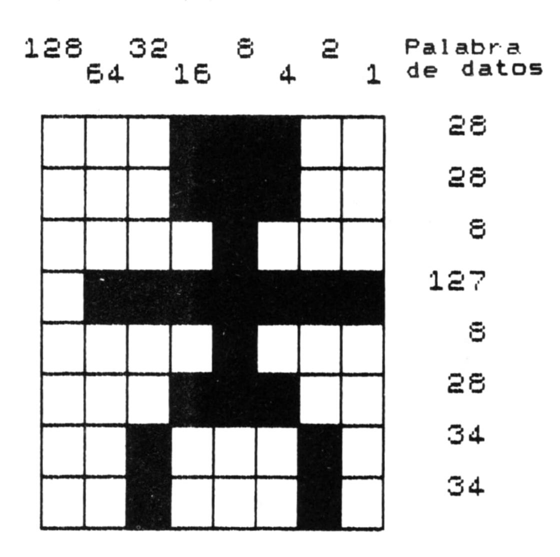


Fig. 5.1. Conversión de patrones de puntos en números.

Para introducir en la memoria el patrón de puntos, tenemos que escribir la secuencia de ocho números en ocho posiciones sucesivas de memoria. Lo podemos hacer utilizando el comando POKE, que toma la forma:

POKE dirección, valor

donde dirección es la dirección real que tiene en la memoria del ordenador, y el valor es el número que queramos escribir en esa dirección.

Todo lo que tenemos que hacer es colocar las palabras de datos en el lugar adecuado de la memoria. Afortunadamente, no necesitamos conocer la dirección real, ya que el ordenador puede buscarla. Suponga que queremos colocar nuestro patrón de puntos en el primer espacio de símbolo disponible. Recordará que inicialmente será el símbolo A. Para colocar en él los datos de la primera fila, podemos usar:

POKE USR“a”, datos

donde “datos” es el valor de la fila superior de puntos. Para colocar la siguiente fila de puntos, podemos utilizar:

POKE USR“a”+1,datos

y “datos” es ahora el número de la segunda fila de puntos. Podemos continuar del mismo modo para las siguientes filas de puntos.

Una vez obtenido el patrón del símbolo, ¿cómo podemos visualizarlo en la pantalla? Podemos utilizar una instrucción PRINT con un término CHR$, como antes hicimos con los símbolos gráficos de mosaico. Los códigos de carácter para los símbolos definidos por el usuario van de 144 a 164. Si elegimos USR“a” para fijar la dirección de POKE, el código de carácter será 144. En la figura 5.2 se listan los códigos de dirección USR y el código de carácter asociado para los esquemas de puntos que seleccionan.

Letra USR	Código de carácter de símbolo a medida	Letra USR	Código de carácter de símbolo a medida
A	144	L	155
B	145	M	156
C	146	N	157
D	147	O	158
E	148	P	159
F	149	Q	160
G	150	R	161
H	151	S	162
I	152	T	163
J	153	U	164
K	154		

Fig. 5.2. Códigos de dirección USR y los correspondientes códigos de símbolo definidos por el usuario.

Otra forma de imprimir los símbolos especiales es utilizar el teclado en su modo gráfico. Esto se consigue pulsando CAPS SHIFT y la tecla nueve a la vez, que harán cambiar al cursor a la modalidad G.

En lo que se refiere a los símbolos de mosaico, pueden utilizarse las teclas que llevan los dibujos de esos símbolos, y para los símbolos definidos por el usuario bastará con teclear una de las letras de la A a la U, según se desee un símbolo u otro.

Hay otra forma para definir los esquemas de puntos para un símbolo. Para ello utilizaremos la palabra binaria (cadena de "1" y "0"). Para que el ordenador sepa que los datos están en forma binaria, ponemos delante de la cadena de bits de datos en binario la instrucción BIN. Así pues, la fila superior de nuestro monigote podría escribirse:

BIN ØØØ111ØØ

Observe que BIN es una palabra de instrucción, y se obtiene presionando la tecla B cuando el teclado está operando en el modo extendido (cusor E).

El programa listado de la figura 5.3 utiliza este método para especificar el patrón de puntos.

```
100 REM Preparacion de simbolos especiales
110 REM utilizando BIN.
120 POKE USR "a",BIN 00011100
130 POKE USR "a"+1,BIN 00011100
140 POKE USR "a"+2,BIN 000010000
150 POKE USR "a"+3,BIN 01111111
160 POKE USR "a"+4,BIN 00001000
170 POKE USR "a"+5,BIN 00011100
180 POKE USR "a"+6,BIN 00100010
190 POKE USR "a"+7,BIN 00100010
200 FOR n=1 TO 20
210 LET r=INT (RND*20)
220 LET c=INT (RND*30)
230 PRINT AT r,c;CHR$ 144;
240 NEXT n
```

Fig. 5.3. Utilización del comando BIN para la creación de patrones de puntos.

Patrones más complejos

Algunas veces podemos encontrar que la forma que deseamos producir para nuestro símbolo es tan compleja que no puede visuali-

zarse con una matriz de puntos de 8 × 8. Este problema puede solucionarse fácilmente, realizando el patrón a partir de un grupo de símbolos especiales, e imprimiendo los grupos uno al lado del otro. El principio seguido es el mismo que utilizamos en el Capítulo Dos cuando creamos el monigote a partir de un patrón de símbolos de mosaico de 4 × 4.

Un método alternativo se muestra en el programa listado en la figura 5.4. En él, se ha formado una matriz de dos dimensiones con 8 filas y 8 columnas. Cada número de la matriz se puede fijar en 1 o en 0, según se desee o no un punto en ese lugar en el patrón de puntos.

```
100 REM Preparacion de matrices de puntos
110 REM para simbolos o manchas.
120 DIM d(8,8)
125 REM Preparacion de la matriz
130 FOR b=1 TO 8
140 FOR a=1 TO 8
150 READ d(a,b)
160 NEXT a
170 NEXT b
180 DATA 0,0,0,1,1,1,0,0
190 DATA 0,0,0,1,1,1,0,0
200 DATA 0,0,0,0,1,0,0,0
210 DATA 0,1,1,1,1,1,1,1
220 DATA 0,0,0,0,1,0,0,0
230 DATA 0,0,0,1,1,1,0,0
240 DATA 0,0,1,0,0,0,1,0
250 DATA 0,0,1,0,0,0,1,0
260 FOR n=1 TO 20
270 LET x=INT (RND*200)
280 LET y=10+INT (RND*150)
285 REM Dibujo del simbolo.
290 FOR b=0 TO 7
300 FOR a=0 TO 7
310 IF d(a+1,b+1)=0 THEN GO TO 330
320 PLOT x+a,y-b
330 NEXT a
340 NEXT b
350 NEXT n
```

Fig. 5.4. Utilización de una matriz para almacenar un patrón de puntos de un monigote.

Para visualizar el símbolo, se utilizan dos bucles de recuento que analizan todos los valores de d. Al mismo tiempo, el cursor gráfico se mueve por la pantalla para localizar el patrón de puntos. En cada paso se comprueba el valor de d para ver si es un "1" o un "0". Cuando d es un "1", se traza un punto en la pantalla, pero cuando es un "0", la instrucción plot se salta, y el programa sigue el proceso de buscar la posición del siguiente punto.

Esta técnica de producción de formas utiliza mucha memoria, pero posiblemente sea más sencilla de organizar y más flexible, para crear la forma, que el uso de una matriz de símbolos especiales.

El comando POINT

En algunos casos necesitamos conocer el estado de un punto de la pantalla. Esto se puede conseguir muy fácilmente utilizando el comando POINT. Esta palabra de instrucción se obtiene seleccionando el modo de teclado extendido (aparecerá el cursor E con intermitencia), y pulsando luego la tecla 8 con SYMBOL SHIFT.

El comando POINT completo toma la forma siguiente:

```
1ØØ  LETp = POINT(x,y)
```

donde x e y son las coordenas de posición del punto que deseamos comprobar. Si el punto de la pantalla está con el color INK, el valor resultante para p será 1, si el punto está con el color PAPER (de fondo), p tendrá un valor 0.

Aunque POINT nos dice si el punto elegido en la pantalla está activado (on) o no, (off), no nos dice el color de ese momento, y para saberlo necesitamos encontrar los atributos de color del espacio de carácter que contiene el punto a que nos estamos refiriendo. Esto puede hacerse utilizando la instrucción ATTR, que examinaremos en el Capítulo Seis.

La instrucción POINT puede utilizarse también en una instrucción IF. Veámoslo:

```
1ØØ IF POINT(x,y) = 1 THEN GOTO 2ØØ
```

En este caso, si el punto está activado, el resultado del comando IF es "verdad", y el programa saltará a la línea 2ØØ. Si el punto no está activado, el programa continúa con la siguiente instrucción, ya que el resultado del comando IF es "falso". Si desea que el programa salte cuando el punto no está activado, cambie la instrucción IF, como sigue:

```
1ØØ IF POINT(x,y) = Ø THEN GOTO 2ØØ
```

Ahora utilizaremos el comando POINT para producir algunas manipulaciones interesantes con los patrones de los símbolos que se visualizan.

Colocación de símbolos en la pantalla de alta resolución

El método más sencillo de insertar texto en una pantalla de alta resolución es utilizar la instrucción PRINT AT con la que el símbolo se imprime directamente en un punto específico de la pantalla. Esta técnica es muy adecuada para muchos trabajos, pero sólo sirve para colocar símbolos en las posiciones normales de los símbolos en la pantalla. Sin embargo, habrá ocasiones en que queramos colocar un símbolo en un punto específico de la pantalla que quede entre las posiciones normales de impresión. Esto se consigue fácilmente, y vamos a examinar a continuación el modo de realizarlo.

Recuerde que un símbolo consiste en un conjunto de puntos que pueden colocarse en un punto cualquiera de la pantalla sin problemas, utilizando el comando PLOT. Supongamos que queremos tomar el símbolo A y colocarlo en un punto aleatorio de la pantalla. Lo primero que necesitamos es saber el patrón de puntos que compone el símbolo A. La manera más sencilla de saberlo es imprimir el símbolo A en la posición Ø,Ø de la pantalla. Sabiendo la posición exacta de este patrón de puntos podemos utilizar el comando POINT para descubrir qué puntos están con el color INK (de texto) y qué puntos están con el color PAPER (de fondo). Con una sencilla operación de bucle podemos examinar cada punto del símbolo impreso, en un momento dado, para poder escribir una copia del símbolo en cualquier lugar de la pantalla que deseemos.

El programa listado en la figura 5.5 muestra cómo con esta técnica puede imprimirse el símbolo A, en un punto cerca del centro de la pantalla.

```
100 REM Coloca un simbolo en el centro de la
pantalla
110 REM copiando sus puntos
120 PRINT AT 0,0;"A";
130 LET x=128
140 LET y=88
150 REM Copia el patron de puntos
160 FOR b=0 TO 7
170 FOR a=0 TO 7
180 IF POINT (a,175-b)=0 THEN GO TO 200
190 PLOT x+a,y-b
200 NEXT a
210 NEXT b
```

Fig. 5.5. Transferencia de un símbolo por copia de puntos.

El símbolo elegido se imprime primero en el ángulo superior izquierdo de la pantalla, para mostrar el patrón de puntos que

vamos a copiar. La fila superior de puntos de este patrón tiene x posiciones que van desde Ø hasta 7, y su coordenada y es 175.

La siguiente fila de puntos tiene también las mismas posiciones de x, pero, en este caso, y vale 174.

Las filas sucesivas son iguales, excepto que la coordenada y se va reduciendo una unidad por cada fila. Para localizar los puntos podemos establecer dos bucles de recuento con variables a y b, ambas dentro de una gama de Ø a 7.

En cuanto al comando POINT, los parámetros x e y serán a, y 175—b, y nosotros sólo tendremos que examinar si el resultado es 1 o no, para saber si el punto tiene un color INK. Una vez examinado el esquema de puntos, tendremos que trazar una copia de este patrón de puntos en otro lugar de la pantalla que elijamos. En primer lugar debemos definir el punto en el que queremos trazar el símbolo, y las coordenadas x, y de ese punto se tomarán como extremo superior izquierdo del símbolo que va a ser dibujado.

Ahora, para trazar los puntos, usaremos sencillamente PLOT x+a,y—b, donde a y b son las mismas variables de recuento que usamos en la instrucción POINT. Si la prueba de POINT muestra un punto iluminado, se dibuja un punto en la nueva posición del símbolo, pero, si el punto no está iluminado, se salta la instrucción PLOT.

A medida que progresan los dos bucles, el patrón de puntos se copiará desde la esquina superior izquierda a la nueva posición. Utilizando esta rutina, es fácil sobreponer dos símbolos. Puede verlo demostrado en el programa listado en la figura 5.6., donde el símbolo se copia en posiciones aleatorias por toda la pantalla.

```
100 REM Colocacion de un simbolo en
posicion aleatoria
110 REM mediante la copia de puntos
120 PRINT AT 0,0;"A";
130 FOR n=1 TO 50
140 LET x=8+INT (RND*240)
150 LET y=8+INT (RND*160)
160 GO SUB 400
170 NEXT n
180 STOP
390 REM Copia el simbolo en el punto x,y.
400 FOR b=0 TO 7
410 FOR a=0 TO 7
420 IF POINT (a,175-b)=0 THEN GO TO 440
430 PLOT x+a,y-b
440 NEXT a
450 NEXT b
460 RETURN
```

Fig. 5.6. Transferencia de símbolos a posiciones aleatorias en la pantalla.

Se producirá un dibujo en la pantalla, semejante al que se muestra en la figura 5.7. Observe que en este caso se pueden tener símbolos uno sobre el otro. Este método de dibujar símbolos de texto en la pantalla es particularmente útil para insertar texto en un dibujo gráfico de alta resolución.

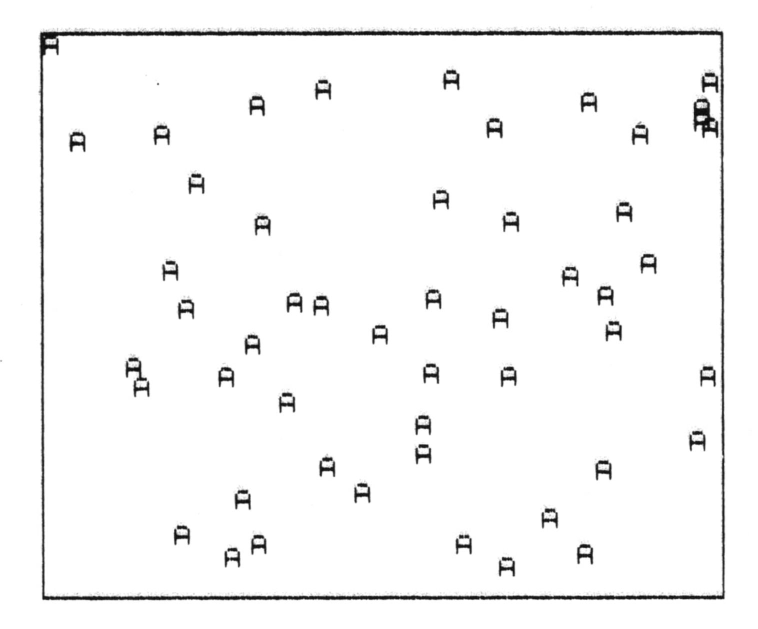


Fig. 5.7. Visualización típica de símbolos en posiciones aleatorias.

Rotación de los símbolos

Cambiando ligeramente el bucle de copia, podemos producir algunos efectos interesantes. Supongamos que queremos escribir el símbolo al revés en la pantalla. Si cambiamos la coordenada y de la instrucción PLOT a y+b, se invertirá efectivamente el símbolo, ya que la fila superior de puntos que se leyó del patrón maestro ahora es la fila inferior de puntos del símbolo que hemos trazado. La secuencia de todas las otras filas también cambia. Un punto importante a notar es que el símbolo trazado no tendrá su vértice inferior izquierdo en la posición x,y especificada. La posición x,y debe elegirse de forma que la coordenada y de la fila superior del símbolo trazado no vaya más allá de 175, ya que, si no es así, el programa se parará dando error.

Si usted quiere el símbolo al revés (boca abajo), pero con su

vértice superior izquierdo como antes en x,y, deberá cambiar el comando PLOT a:

PLOT x+a,y—8+b

Esto le permitirá insertar fácilmente el símbolo en una fila impresa de símbolos de texto.

Para colocar un símbolo simétrico del anterior, frente a él, para que parezca la imagen de un espejo, podemos aplicar una técnica semejante a la coordenada x del comando PLOT, de forma que se convertirá en

PLOT x+8—a,y—b

Seamos ahora algo más intrépidos y veamos qué pasa si cambiamos uno por otro los términos de copia a y b en el comando PLOT, es decir , hagamos

PLOT x+b,y+a

Esta expresión produce un símbolo que ha girado sobre su lado, ya que las filas horizontales del patrón original se han trazado en vertical. Para volver el símbolo hacia el otro lado, tendremos que cambiar simplemente el comando

PLOT x—b,y+a

```
100 REM Rotacion de simbolos de texto
110 REM mediante la copia de puntos
120 PRINT AT 0,0;"A";
130 LET x=124
140 LET y=96
150 GO SUB 400
160 LET x=124
170 LET y=64
180 GO SUB 500
190 LET x=112
200 LET y=76
210 GO SUB 600
220 LET x=144
230 LET y=84
240 GO SUB 700
250 PRINT AT 0,0;" ";
260 STOP
390 REM Simbolo normal
400 FOR b=0 TO 7
410 FOR a=0 TO 7
420 IF POINT (a,175-b)=0 THEN GO TO 440
430 PLOT x+a,y-b
440 NEXT a
450 NEXT b
```

```
460 RETURN
490 REM Simbolo invertido.
500 FOR b=0 TO 7
510 FOR a=0 TO 7
520 IF POINT (a,175-b)=0 THEN GO TO 540
530 PLOT x+a,y+b
540 NEXT a
550 NEXT b
560 RETURN
590 REM Simbolo girado a la izquierda.
600 FOR b=0 TO 7
610 FOR a=0 TO 7
620 IF POINT (a,175-b)=0 THEN GO TO 640
630 PLOT x+b,y+a
640 NEXT a
650 NEXT b
660 RETURN
690 REM Simbolo girado a la derecha.
700 FOR b=0 TO 7
710 FOR a=0 TO 7
720 IF POINT (a,175-b)=0 THEN GO TO 740
730 PLOT x-b,y-a
740 NEXT a
750 NEXT b
760 RETURN
```

Fig. 5.8. Rotación de símbolos por copia de puntos.

Una vez más debemos hacer una corrección en los valores básicos x e y de estos comandos, para que el ángulo superior izquierdo del nuevo símbolo siga en el punto x,y de la pantalla.

El programa que se lista en la figura 5.8 dibuja símbolos en las cuatro orientaciones, y produce un dibujo similar al que se muestra en la figura 5.9.

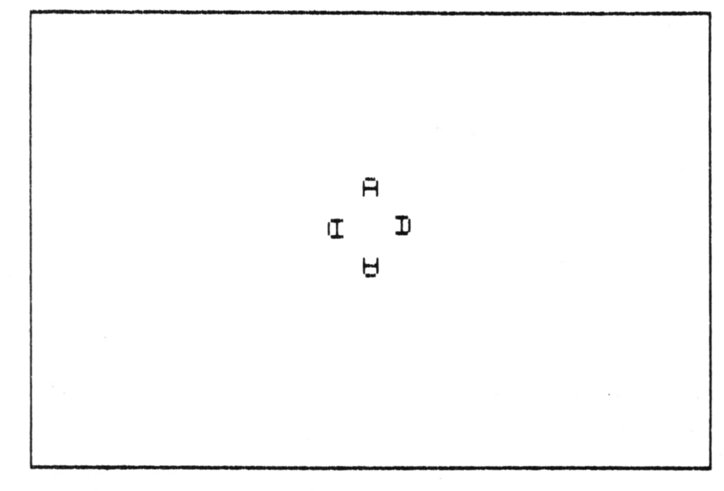


Fig. 5.9. Visualización de símbolos girados.

Carácteres más grandes y mejores

Suponga que queremos producir en la pantalla un símbolo de doble anchura. Para conseguir la doble anchura podemos usar simplemente un segundo comando PLOT que coloque un punto al lado del primero. Por supuesto, deberemos tener dos espacios en blanco por cada punto en blanco. Todo esto puede conseguirse utilizando un copiador doble, que dibuja trazando dos puntos por cada punto del patrón original.

```
100 REM Produccion de simbolos grandes.
110 LET cy=165
115 REM Altura del simbolo = v
120 FOR v=1 TO 5
130 LET cx=10
140 LET cy=cy-8*v
145 REM Ancho del simbolo = h
150 FOR h=1 TO 5
160 PRINT AT 0,0;"$";
170 LET cx=cx+10*h
180 GO SUB 500
190 NEXT h
200 NEXT v
220 PRINT AT 0,0;" ";
230 STOP
490 REM Subrutina de copiado del simbolo.
500 FOR y=0 TO 7
510 FOR x=0 TO 7
520 IF POINT (x,175-y)=0 THEN GO TO 580
530 FOR k=0 TO v-1
540 FOR j=0 TO h-1
550 PLOT cx+h*x+j,cy-v*y+k
560 NEXT j
570 NEXT k
580 NEXT x
590 NEXT y
600 RETURN
```

Fig. 5.10 Programa que produce símbolos mayores.

Es igual de sencillo generar símbolos de doble altura. En este caso se aplica el proceso al copiador b en lugar de hacerlo al a. Ahora cada fila de puntos en el patrón original se inspecciona dos veces y produce dos filas de puntos, una sobre la otra, en el carácter copia.

Si combinamos las dos acciones, podemos producir símbolos de doble tamaño. Si seguimos desarrollando estas posibilidades, podemos obtener tamaños triples o cuádruples partiendo del patrón de puntos original. Hay que observar, sin embargo, que, en todas la

rutinas de copia de patrones de puntos, ninguno de ellos se salga fuera de los límites de la pantalla.

El programa listado en la figura 5.10 nos da una idea de las posibilidades de esta técnica, y nos muestra una gama de símbolos cuyos tamaños se han aumentado hasta cinco veces el tamaño original. El dibujo que ofrece este programa sobre la pantalla puede verse en la figura 5.11. Esta técnica puede resultar muy útil para establecer títulos en negrita sobre la pantalla.

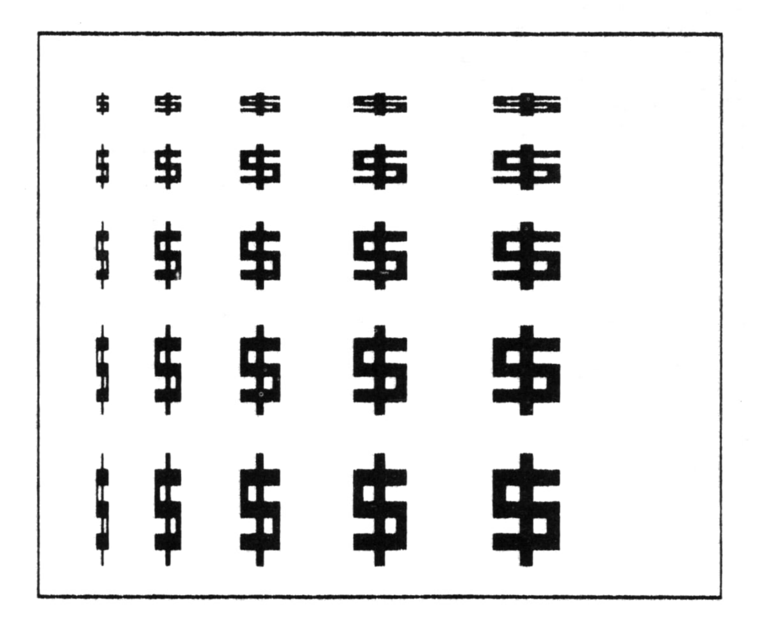


Fig. 5.11. Dibujo de símbolos aumentando su tamaño.

Símbolos con inclinación

Algunas veces desearemos producir símbolos en itálica, es decir, símbolos que no son verticales, sino inclinados un cierto ángulo. Este resultado se obtiene restando la copia b, del término combinado a y x. En este caso, cada fila de puntos sucesiva está desplazada una posición a la izquierda, de forma que el símbolo se traza con un ángulo, y tiene el aspecto de un símbolo en itálica exagerado. Utilizando INT (b/2) en lugar de b, produciremos un símbolo en itálica más real, ya que el ángulo del símbolo será menor.

Si en lugar de sutraer el término b, éste se añade al término x, los símbolos se inclinarán en la otra dirección. El resultado de este tipo

de operaciones puede verse ejecutando el programa listado en la figura 5.12., que demuestra este efecto, produciendo gran variedad de símbolos y formas.

```
100 REM Produccion de simbolos grandes en
italica.
110 LET cy=165
115 REM Altura del simbolo = v
120 FOR v=1 TO 5
130 LET cx=10
140 LET cy=cy-8*v
145 REM Ancho del simbolo = h
150 FOR h=1 TO 5
160 PRINT AT 0,0;"$";
170 LET cx=cx+10*h
180 GO SUB 500
190 NEXT h
200 NEXT v
220 PRINT AT 0,0;"  ";
230 STOP
490 REM Subrutina de copiado del simbolo.
500 FOR y=0 TO 7
510 FOR x=0 TO 7
520 IF POINT (x,175-y)=0 THEN GO TO 580
530 FOR k=0 TO v-1
540 FOR j=0 TO h-1
550 PLOT cx+h*x+j-y*h/2,cy-v*y+k
560 NEXT j
570 NEXT k
580 NEXT x
590 NEXT y
600 RETURN
```

Fig. 5.12. Creación de símbolos de itálica.

Otras posibilidades con las que puede experimentar son añadir la copia a al término y, que dará un carácter con líneas horizontales inclinadas, o bien combinar ambas operaciones, lo que dará un carácter girado 45 grados. Como esta técnica de rotación no sigue las ecuaciones normales de rotación, con este método la forma del símbolo quedará algo distorsionada. Dejo al lector que experimente con las posibles combinaciones que pueden aplicarse al trazado de un carácter copiado.

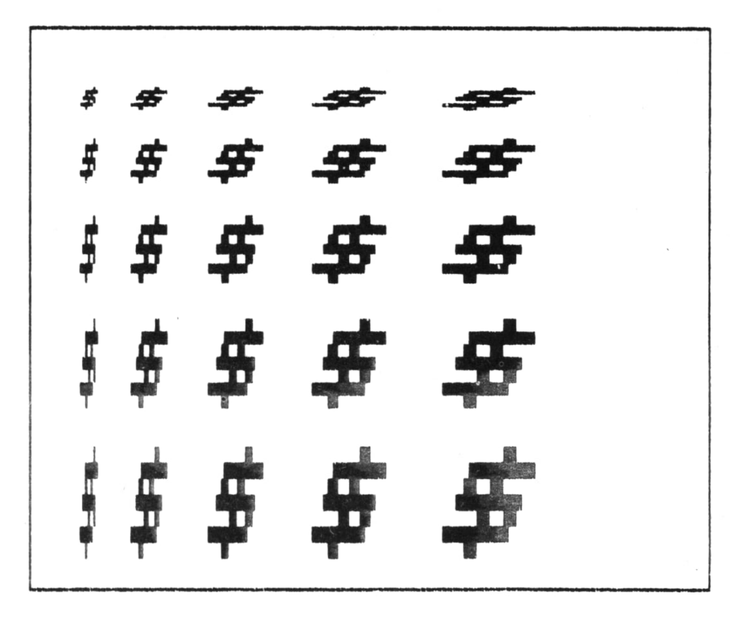


Fig. 5.13. Visualización de símbolos en itálica.

Capítulo 6

Más sobre el color

Hasta el momento, en este libro, hemos usado los comandos INK y PAPER para establecer los colores de dibujo (color de escritura o color INK y de fondo, o color PAPER). Existen muchos otros comandos relacionados con el color en pantalla, y exploraremos algunos de ellos para ver cómo pueden obtenerse muchos más de los ocho colores standard de los comandos INK y PAPER.

Colores brillantes e intermitentes

Algunas veces desearemos resaltar determinadas partes de un texto, que se está visualizando, para atraer la atención del lector. Un ejemplo puede ser advertir alguna situación peligrosa potencial, o indicar alguna acción que deba tomarse. Por ejemplo, el Spectrum, en el sistema de visualización de texto, tiene un cursor intermitente que indica la posición del siguiente símbolo que se imprimirá.

Una forma de resaltar el texto es utilizar el comando BRIGHT. Este comando puede llevar a continuación un 1 ó un Ø. El comando BRIGHT 1 hace que cualquier nuevo símbolo en la pantalla sea más brillante de lo normal, pero también establecerá que el color de fondo de estos símbolos sea más brillante. El comando BRIGHT no tiene efecto sobre símbolos o fondos negros. Para restaurar la visualización normal, se utiliza el comando BRIGHT Ø. Obsérvese que BRIGHT sólo afecta a los símbolos que se han impreso después de haber usado el comando BRIGHT 1.

Algunas veces, dependiendo del receptor de televisión que se utilice para la visualización, BRIGHT 1 causará la eliminación del brillo por todo el dibujo.

Otro sistema de llamar la atención en una zona particular de la pantalla, es utilizar el comando FLASH. Cuando se usa FLASH,

todos los nuevos caracteres que se escriban serán intermitentes. De hecho, lo que ocurre es que los colores de fondo y de texto dentro del espacio del carácter se alternan, de forma que, en un momento, nos encontramos con un símbolo negro sobre fondo blanco, y en otro, un símbolo blanco sobre un fondo negro. Como ocurría con el comando BRIGHT, si insertamos un comando FLASH 0, los siguientes símbolos se visualizarán en los colores normales, de antes. Después de FLASH 0, ningún símbolo seguirá intermitente, si no se escriben de nuevo con otro comando FLASH 1.

El programa que se lista en la figura 6.1. demuestra el efecto que producen los comandos BRIGHT y FLASH.

```
100 REM Simbolos brillantes e intermitentes.
110 FOR k=1 TO 4
120 PRINT AT k,0;" ";
130 FOR j=1 TO 7
140 INK j
150 PRINT CHR$ (136+j);
160 PRINT CHR$ (136+j);
170 PRINT CHR$ (136+j);
180 NEXT j
190 NEXT k
200 FOR k=5 TO 8
210 BRIGHT 1
220 PRINT AT k,0;" ";
230 FOR j=0 TO 7
240 INK j
250 PRINT CHR$ (136+j);
260 PRINT CHR$ (136+j);
270 PRINT CHR$ (136+j);
280 NEXT j
290 NEXT k
300 BRIGHT 0
310 FOR k=9 TO 12
320 FLASH 1
330 PRINT AT k,0;" ";
340 FOR j=0 TO 7
350 INK j
360 PRINT CHR$ (136+j);
370 PRINT CHR$ (136+j);
380 PRINT CHR$ (136+j);
390 NEXT j
400 NEXT k
410 FLASH 0
420 INK 0
430 PRINT AT 2,25;"Normal";
440 PRINT AT 6,25;"Brillan";
445 PRINT AT 7,25;"te";
450 PRINT AT 10,25;"Intermi";
460 PRINT AT 11,25;"tente";
```

Fig. 6.1. Demostración de los comandos BRIGHT y FLASH.

Figuras coloreadas

Una forma de obtener dibujos de color con más cuerpo, es visualizar bloques o zonas de color en lugar de líneas o puntos. Esto puede hacerse rellenando una zona de la pantalla, de forma que todos los puntos de esa zona tengan el color INK.

Tomemos una figura muy simple, como un rectángulo, por ejemplo. Para llenar un rectángulo, podemos comenzar por dibujar el lado inferior, y después ir trazando líneas sucesivamente una sobre la otra, y con la longitud de todas ellas igual a la anchura del rectángulo, hasta llegar a dibujar el lado superior.

```
100 REM Rectangulo relleno con color.
110 FOR n=1 TO 20
120 CLS
130 INK INT (RND*7)
140 LET w=15+INT (RND*100)
150 LET x=INT (RND*(255-w))
160 LET h=10+INT (RND*60)
170 LET y=INT (RND*(175-h))
180 PLOT x,y
190 DRAW w,0
200 DRAW 0,h
210 DRAW -w,0
220 DRAW 0,-h
230 IF h>w THEN GO TO 290
235 REM Llena con cintas horizontales.
240 FOR j=1 TO h
250 PLOT x,y+j
260 DRAW w,0
270 NEXT j
280 GO TO 330
285 REM Llena con lineas verticales.
290 FOR j=1 TO w
300 PLOT x+j,y
310 DRAW 0,h
320 NEXT j
330 PAUSE 50
340 NEXT n
```

Fig. 6.2. Programa que produce rectángulos con color.

En el programa que se lista en la figura 6.2., las líneas sucesivas se van dibujando por el sistema de trazar primero un punto y luego dibujar una línea horizontal igual a la anchura del rectángulo. Para la siguiente línea, la coordenada y se incrementa en 1, y la acción se repite. El proceso de llenado se lleva a cabo en un bucle que se ejecuta h veces, siendo h la altura del rectángulo.

Hay dos modos de llenar un círculo. El primer método consiste en dibujar una serie de círculos concéntricos, cuyo radio se va incrementando una unidad hasta alcanzar el radio deseado. Puede ver este sistema en el programa listado en la figura 6.3.

```
100 REM Rellenado de circulos
110 REM variando el radio.
120 FOR n=1 TO 25
130 LET r=INT (RND*50)
140 LET x=r+INT (RND*(255-2*r))
150 LET y=r+INT (RND*(175-2*r))
160 INK INT (RND*7): CLS
165 REM Dibuja la circunferencia.
170 CIRCLE x,y,r
175 REM Llena el circulo.
180 FOR j=0 TO r
190 CIRCLE x,y,j
200 NEXT j
210 PAUSE 50
220 NEXT n
```

Fig. 6.3. Programa para colorear un círculo variando el radio.

```
100 REM Rellenado de circulos mediante lineas
radiales.
120 FOR n=1 TO 25
130 LET r=INT (RND*50)
140 LET x=r+INT (RND*(255-2*r))
150 LET y=r+INT (RND*(175-2*r))
160 INK INT (RND*7): CLS
165 REM Dibuja la circunferencia.
170 CIRCLE x,y,r
175 REM Llena el circulo.
180 LET dt=PI/(r*4)
190 FOR j=0 TO r*8
200 PLOT x,y
210 DRAW r*COS (j*dt),r*SIN (j*dt)
220 NEXT j
230 PAUSE 50
240 NEXT n
```

Fig. 6.4. Coloreado de un círculo utilizando líneas radiales.

Otra alternativa para llenar un círculo es dibujar líneas radiales desde el centro, e ir incrementando el ángulo en pequeñas cantidades, de forma que todos los puntos del círculo se llenen. Este sistema puede verse en el programa de la figura 6.4.

El llenado de un polígono regular, como, por ejemplo, un hexágono, se lleva a cabo mejor dibujando series de polígonos concéntricos, con un radio que se va incrementando paso a paso hasta alcanzar el tamaño deseado.

Diferentes tonalidades de color

Con los comandos INK y PAPER, pueden obtenerse normalmente ocho colores diferentes. Esta gama puede extenderse, sin embargo, utilizando el comando BRIGHT, cuyo efecto añade blanco al color elegido, produciendo de este modo un símbolo o punto más brillante en la pantalla, y una tonalidad de color más pálida que la básica. Así pues, BRIGHT aplicado a un símbolo rojo, producirá un símbolo rosa.

```
100 REM Mezclas de colores utilizando lineas
horizontales.
120 FOR p=7 TO 0 STEP -1
130 FOR i=0 TO 7
140 PAPER 7
150 CLS
160 INK 0
170 PAPER p
180 PLOT 87,47
190 DRAW 82,0
200 DRAW 0,82
210 DRAW -82,0
220 DRAW 0,-82
230 INK i
240 FOR j=0 TO 77 STEP 2
250 PLOT 88,50+j
260 DRAW 79,0
270 PLOT 88,49+j
280 DRAW PAPER p; INVERSE 1;79,0
290 NEXT j
300 PAUSE 50
310 NEXT i
320 NEXT p
```

Fig. 6.5. Mezcla de colores utilizando líneas horizontales.

Es evidente que pueden obtenerse muchos más colores mezclando en la pantalla los colores básicos.

Si dibuja una línea roja junto a una línea amarilla, el resultado parece naranja, porque, a la distancia normal del espectador, las líneas adyacentes tienden a mezclarse.

Si tomamos un cuadrado y lo coloreamos, pero dibujamos las líneas alternando dos colores, habremos obtenido una forma de mezclar colores. El programa que se lista en la figura 6.5. muestra el tipo de resultados y las diferentes tonalidades de color que se pueden obtener.

```
100 REM Mezcla de colores
110 REM utilizando lineas verticales alternas.
120 FOR p=7 TO 0 STEP -1
130 FOR i=0 TO 7
140 PAPER 7
150 CLS
160 INK 0
170 PAPER p
180 PLOT 87,47
190 DRAW 82,0
200 DRAW 0,82
210 DRAW -82,0
220 DRAW 0,-82
230 INK i
240 FOR j=0 TO 77 STEP 2
250 PLOT 90+j,48
260 DRAW 0,79
270 PLOT 89+j,48
280 DRAW PAPER p; INVERSE 1;0,77
290 NEXT j
300 PAUSE 50
310 NEXT i
320 NEXT p
```

Fig. 6.6. Mezcla de colores utilizando líneas verticales.

Las mezclas de color pueden llevarse a cabo utilizando también tiras de colores alternos que llenen el cuadrado. Este sistema puede verse en el programa listado en la figura 6.6. Usted, sin embargo, encontrará que en muchas combinaciones aparece una serie de líneas en el cuadrado, y que los colores producidos cambian continuamente dando un efecto de parpadeo. El resultado es mejor si se utilizan líneas horizontales y verticales, como se muestra en el programa que se lista en la figura 6.7.

```
100 REM Mezcla de colores utilizando lineas
entrecruzadas.
120 FOR p=7 TO 0 STEP -1
130 FOR i=0 TO 7
140 PAPER 7
150 CLS
160 INK 0
170 PAPER p
180 PLOT 87,47
190 DRAW 82,0
200 DRAW 0,82
210 DRAW -82,0
220 DRAW 0,-82
230 INK i
235 REM Dibuje lineas horizontales.
240 FOR j=0 TO 79 STEP 2
250 PLOT 88,49+j
260 DRAW 78,0
```

```
270 NEXT j
275 REM Dibuje lineas verticales.
280 FOR j=0 TO 77 STEP 2
290 PLOT PAPER p; INVERSE 1;89+j,49
300 DRAW PAPER p; INVERSE 1;0,77
310 NEXT j
320 PAUSE 50
330 NEXT i
340 NEXT p
```

Fig. 6.7. Mezcla de colores utilizando líneas entrecruzadas.

La mezcla de colores utilizando líneas alternas es muy pesada. Una técnica mejor consiste en producir un patrón de puntos alternos. Pueden usarse para ello los símbolos gráficos definidos por el usuario. Con ellos se formará un carácter especial, que contenga, por ejemplo, un patrón de ajedrez a base de puntos, como puede verse en la figura 6.8.

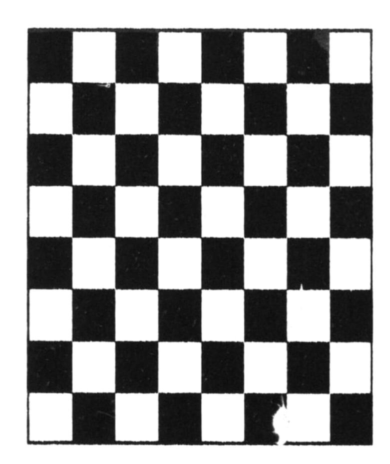


Fig. 6.8. Patrón de tablero de ajedrez para la mezcla de colores.

En este caso, tanto los puntos verticales como los horizontales se alternan en colores INK y PAPER, de forma que en el interior del espacio del símbolo los dos colores estarán mezclados con toda efectividad. Esta técnica se utiliza en el programa que se muestra en la figura 6.9., y que produce una gama amplia de tonalidades de color.

```
100 REM Mezcla de colores
110 REM por impresion de simbolos entremezclados.
```

```
120 REM Crea el simbolo.
130 FOR n=0 TO 7 STEP 2
140 POKE USR "a"+n,170
150 POKE USR "a"+n+1,85
160 NEXT n
170 FOR p=7 TO 0 STEP -1
180 PAPER p
190 FOR i=0 TO 7
200 INK i
210 FOR b=0 TO 1
220 BRIGHT b
230 FOR r=0 TO 15
240 FOR c=10 TO 20
250 PRINT AT r,c;CHR$ 144;
260 NEXT c
270 NEXT r
280 PAUSE 50
290 NEXT b
300 BRIGHT 0
310 NEXT i
320 NEXT p
```

Fig. 6.9. Mezcla de colores mediante la utilización de símbolos especiales.

Visualización invertida

Los símbolos van escritos normalmente en el color de texto INK sobre un color de fondo PAPER. Si utiliza el comando INVERSE 1, los símbolos se visualizarán en el color PAPER sobre un fondo de color INK. Esto puede resultar útil para destacar ciertas palabras, en un mensaje. También podemos hacer que ciertas palabras tengan intermitencia, usando alternativamente la instrucción INVERSE 1 e INVERSE Ø. Sin embargo, hay un método más sencillo de obtener la intermitencia de ciertos símbolos, que es la utilización del comando FLASH.

El comando OVER

Un comando utilísimo para el color en el Spectrum, es el comando OVER. De igual modo que los comandos FLASH e INVERSE, tiene dos modos posibles, OVER Ø y OVER 1.

Cuando se selecciona OVER 1, se añade en ese espacio de carácter un nuevo punto o símbolo a lo que anteriormente existía. La acción del comando OVER es una operación OR exclusiva, entre los nuevos datos que se escriben en la pantalla y los ya existentes. En una operación OR exclusiva, si un punto del nuevo patrón está activado, o el equivalente en la pantalla también está activado, el punto visualizado después de la operación estará activado. Así pues,

si los puntos del nuevo símbolo están en color INK, y el punto de la pantalla estaba en color PAPER, el resultado es un punto en color INK.

Si los dos puntos son del mismo color, ambos INK, o ambos PAPER, la resultante en pantalla es un punto en color PAPER. Obtenemos, pues, la siguiente tabla de resultados:

PUNTO ANTIGUO	PUNTO NUEVO	RESULTADO
PAPER	PAPER	PAPER
INK	PAPER	INK
PAPER	INK	INK
INK	INK	PAPER

El efecto al escribir con OVER dos signos en la misma posición, es el siguiente: si se sobreponen, se escriben con el color PAPER, pero el resto de los puntos en ambos símbolos se visualiza en color INK. Debemos tener en cuenta también que, si ya estaba un signo anterior en un color diferente, sus puntos cambiarán ahora al color INK en curso.

Un efecto útil e interesante ocurre cuando escribimos el mismo símbolo sobre sí mismo, utilizando la función OVER. La primera vez que se escribe el símbolo en un espacio en blanco, se visualizará normalmente en el color INK actual. Cuando vuelve a escribirse el símbolo de nuevo, todos los puntos coinciden con los del símbolo anterior y, por tanto, se escribirán en color PAPER. Como todos los puntos que rellenan el espacio de carácter también están en color PAPER, el resultado es que hemos borrado el carácter.

Este descubrimiento no parece muy inteligente, ya que podíamos obtener el mismo resultado escribiendo un carácter en blanco. Suponga, sin embargo, que, en lugar de comenzar con un espacio en blanco, tenemos un carácter en ese espacio, y que el carácter que queremos escribir es otro distinto. Al escribir el nuevo símbolo la primera vez con el comando OVER, obtendremos una combinación de los dos símbolos. En los puntos que se solapen, los puntos estarán en el color PAPER. Si ahora escribimos otra vez el segundo símbolo, se borrará todo lo que no se solapaba con el símbolo anterior. En los puntos solapados, sin embargo, los puntos estaban en el color PAPER, y ahora estarán en color INK, restaurando el símbolo original en pantalla. Por tanto, hemos obtenido un sistema de escribir y borrar un carácter sobre otro, sin borrar el carácter original.

Realización de un programa de diseño

Hasta ahora, para el dibujo de nuestras líneas, triángulos y rectángulos, hemos tenido que calcular las posiciones de los puntos entre los cuales se van a trazar las líneas. Para ello utilizábamos los parámetros x e y con los comandos PLOT y DRAW, y producíamos una línea en pantalla. En la vida real, tomaremos simplemente un lápiz o una pluma, y dibujaremos la línea donde queramos. En el Spectrum no es muy difícil conseguirlo, y para ello, vamos a producir un pequeño programa de diseño.

En un programa de diseño sencillo, las teclas se pueden utilizar para mover el cursor gráfico por la pantalla. Supongamos ahora que la posición del cursor gráfico representa la posición de la punta de una pluma, y que podemos colocar la pluma con la punta sobre el papel (hacia abajo) o sin apoyar en éste (arriba). Cuando la pluma está hacia abajo, dibujará un punto en la pantalla y, si no se mueve en el estado "abajo", se dibujará una línea siguiendo la trayectoria de la punta de la pluma. Cuando la pluma está hacia arriba, no escribe en pantalla, sino que únicamente cambia de posición. Para controlar la pluma, se utilizan las teclas U (arriba) y D (abajo). Al comienzo del programa, la pluma está en el estado "arriba", y su posición será el ángulo inferior izquierdo de la pantalla, donde X = Ø e Y = Ø.

Surge un programa cuando tenemos la pluma "arriba". Como no se ha dibujado nada, no tenemos forma de saber dónde se encuentra la pluma. En el programa se ha solucionado este problema colocando un punto en el lugar en el que se encuentra la pluma. Cada vez que se mueve la pluma, el punto se borra y se reescribe en la nueva posición, de forma que continuamente nos va mostrando la posición de la pluma.

Si existe un punto encendido, en la posición donde se mueve la pluma, se anota utilizando la instrucción POINT, que guarda el estado de ese punto. Cuando la pluma se mueve a una nueva posición, el estado del punto original se restablece. Si no lo hiciéramos

```
100 REM Programa de dibujo.
110 PRINT "Las teclas de flecha mueven la
pluma hacia arriba/abajo/derecha/izquierda"
120 PRINT "La tecla D baja la pluma,para dibujar"
130 PRINT "La tecla U levanta la pluma,para
permitir el movimiento sin dibujar"
140 PRINT "La tecla E borra la linea"
150 PRINT "Las teclas numericas de 0 a 6
fijan el color"
```

```
160 PRINT "Con las teclas C,V,B y N se obtienen
lineas diagonales"
170 INPUT "Preparado?(S/N)";a$
180 IF a$<>"s" THEN GO TO 170
185 REM Inicializa
190 LET x=0: LET y=0: LET x1=0: LET y1=0
200 LET s=0: LET p=0: INK 0: PAPER 7
210 LET e=0: CLS : PLOT x,y
220 IF INKEY$="" THEN GO TO 220
230 LET a$=INKEY$
235 REM Control de salida
240 IF a$="q" OR a$="Q" THEN STOP
245 REM Comprueba el estado de la pluma.
250 IF a$="u" OR a$="U" THEN LET p=0: GO TO 270
260 IF a$="d" OR a$="D" THEN LET p=1: LET e=0
270 IF a$="e" OR a$="E" THEN LET e=1
275 REM Comprueba las teclas de flecha y fija x,y.
280 IF a$=CHR$ 8 THEN LET x=x-1: GO TO 360
290 IF a$=CHR$ 9 THEN LET x=x+1: GO TO 360
300 IF a$=CHR$ 10 THEN LET y=y-1: GO TO 360
310 IF a$=CHR$ 11 THEN LET y=y+1: GO TO 360
315 REM Comprueba y fija los movimientos
diagonales.
320 IF a$="c" OR a$="C" THEN LET x=x-1: LET
y=y+1
330 IF a$="v" OR a$="V" THEN LET x=x-1: LET
y=y-1
340 IF a$="b" OR a$="B" THEN LET x=x+1: LET
y=y-1
350 IF a$="n" OR a$="N" THEN LET x=x+1: LET
y=y+1
355 REM Comprueba los limites de x e y;ejecuta
la funcion de cierre circular dela pantalla
360 IF x>255 THEN LET x=x-256
370 IF x<0 THEN LET x=x+256
380 IF y>163 THEN LET y=y-164
390 IF y<0 THEN LET y=y+164
395 REM Establece el color.
400 LET c=(CODE a$)-48
410 IF c>6 OR c<0 THEN GO TO 430
420 INK c
425 REM Vuelve al estado anterior
427 REM si la pluma esta levantada.
430 IF p=1 OR s=1 THEN GO TO 450
440 PLOT OVER 1;x1,y1: OVER 0
450 LET s=POINT (x,y)
455 REM Borrado.
460 IF e=1 THEN INVERSE 1
465 REM Dibuja  un nuevo punto.
470 PLOT x,y
475 REM Actualiza el ultimo puntero
480 LET x1=x: LET y1=y
485 REM Imprime la posicion x,y en ese momento.
490 PRINT AT 0,0;"x = ";x;" y = ";y;"    "
500 INVERSE 0
510 GO TO 220
```

Fig. 6.10. Programa de diseño en alta resolución.

de este modo, al pasar la pluma por una línea ya trazada, la borraríamos.

La figura 6.10., ofrece un listado del programa de diseño que utiliza un sistema de control mediante el teclado.

Las flechas y las teclas U y D se controlan continuamente, utilizando un bucle y el comando INKEY$. Si no hay ninguna tecla presionada, INKEY$ devuelve una cadena de espacios en blanco (" ") y la prueba vuelve a repetirse. Si se pulsa una tecla, se introduce a$ en INKEY$, y acto seguido se ve cuál ha sido la tecla pulsada para tomar las medidas oportunas. Para que las teclas de flecha funcionen correctamente, debe presionarse CAPS SHIFT al mismo tiempo que la tecla de la flecha. El programa comprueba el código de la tecla de flecha con el SHIFT pulsado. Si se desea, puede usarse la aproximación utilizada en el programa de diseño del Capítulo Dos. En aquel programa se detectan los números de las teclas que corresponden a las cuatro flechas, de forma que no es necesario utilizar la tecla SHIFT. Si se pulsa una tecla de flecha, actuará la función de autorepetición del teclado Sinclair, y empezará a repetirse una serie de pasos, cuyo resultado final es el dibujo de una línea.

El programa contiene otros refinamientos. Si se pulsa la tecla E, la pluma actuará como un borrador, y borrará todos los puntos sobre los que pase. Esto permite al usuario la corrección de sus errores. La utilización de las flechas permite el dibujo de líneas horizontales y verticales. El programa reconoce también otras cuatro teclas. Estas teclas son las C, V, B y N, que están programadas para dar un movimiento diagonal a la pluma. La tecla N se mueve hacia arriba y hacia la derecha, y la tecla B se mueve hacia abajo y hacia la derecha. Las teclas C y V se mueven hacia la izquierda, y arriba y abajo, respectivamente.

Atributos de color

En el Spectrum, los símbolos de texto se almacenan como patrones de puntos en la memoria principal de video, junto con los patrones de puntos de los dibujos de alta resolución. A diferencia de otros ordenadores, el Spectrum almacena su información en color en una zona separada de la memoria. Así, la imagen de alta resolución se almacena efectivamente como un esquema de puntos blancos y negros.

La información del color no está relacionada con los puntos individuales de la pantalla de alta resolución, sino con los espacios

de caracteres, cada uno de los cuales es una matriz de 8 x 8 ó 64 puntos.

El color de todos los puntos de esta matriz viene determinado por una palabra de datos almacenada en un área separada de la memoria. Esta palabra proporciona los llamados *ATRIBUTOS* de ese espacio de símbolo de la pantalla.

La palabra de atributos contiene 8 bits de datos que se utilizan como se indica en la figura 6.11. Los tres bits inferiores de la palabra de atributos dan el color INK que tendrá el espacio del carácter. Recordará usted, del Capítulo Uno, que los colores se producían combinando los tres colores primarios: rojo, verde y azul. El color se establece activando estos tres bits para el rojo, verde y azul, respectivamente. Si sólo está activado (on) el bit rojo, el símbolo será rojo, pero si están activados el rojo y el verde, el color será amarillo (rojo + verde), y así sucesivamente.

Bit de Datos	Valor Numérico	Atributo de color
0	1	Azul INK
1	2	Rojo INK
2	4	Verde INK
3	8	Azul PAPER
4	16	Rojo PAPER
5	32	Verde PAPER
6	64	BRIGHT
7	120	FLASH

Fig. 6.11. Bits asignados en la palabra de atributos.

Los tres bits siguientes en la palabra están también reservados para el verde, rojo y azul, pero se refieren al color PAPER, y la información no está relacionada con los puntos individuales de la pantalla de alta resolución, sino con el espacio de carácter, consistente en una matriz de 8 x 8, es decir de 64 puntos. El color de todos estos puntos de la matriz viene determinado por una palabra de datos almacenados en una zona separada de la memoria. Esta palabra suministra en la pantalla los atributos de este espacio de símbolo.

Lectura de los atributos

Si queremos saber qué colores INK o PAPER se encuentran en un espacio de carácter, podemos averiguarlo utilizando el comando:

LET a = ATTR(r,c)

donde r y c son los números de fila y de columnas para el espacio de carácter elegido. El resultado es la obtención de los atributos de una posición de pantalla determinada.

En el Spectrum de 48K, las palabras que indican los atributos de los espacios de caracteres en la pantalla, se almacenan en la memoria entre las posiciones 22528 y 23295. Los atributos se almacenan en un cierto orden, comenzando por el atributo de la posición 0, 0 en la posición 22528 de la memoria.

Las siguientes palabras de memoria son atributos de las columnas de la fila 0, y contando de izquierda a derecha. Las otras filas siguen la secuencia. Para localizar un determinado atributo, podemos utilizar el siguiente cálculo:

Dirección de la Memoria = 22528 + (32 x r) = c

Si deseamos saber el color de un espacio determinado, podríamos utilizar el comando PEEK para localizar la posición correspondiente en el área de la memoria de atributos. A continuación, decodificaremos los bits individuales de la palabra, para encontrar los colores INK y PAPER, y las condiciones BRIGHT y FLASH. También podemos establecer una nueva palabra de atributo o alternar la ya existente, utilizando POKE.

Con frecuencia, desearemos comprobar el color de un punto individual en la pantalla de alta resolución. Podemos hacerlo utilizando el comando ATTR en la forma siguiente:

```
1ØØ LET a = ATTR (21 — y/8, x 8)
```

En este caso, para encontrar la posición de la columna, dividiremos simplemente la coordenada x por 8, ya que hay 8 puntos por cada espacio de carácter. El cálculo de las y es más complicado, ya que en el modo gráfico la y aumenta desde la parte inferior de la pantalla a la superior, mientras que en el modo de texto, las filas se numeran comenzando desde la parte superior y bajando. En este

caso, restaremos y/8 del número de filas de texto de que dispongamos, y obtendremos el número de fila.

Podemos ahora decodificar la palabra de atributos, comprobando el estado de cada bit de datos (uno por uno), y fijando una variable o "señalizador" en 1 ó en 0. De este modo, empezamos comprobando si es menor que 128, en cuyo caso el bit de FLASH debe ser 0, y podemos dar a la variable FL el valor 0. Si a >= 128, a FL se le asigna valor 1, y se restará 128 de a, para prepararlo para la próxima comprobación. Se compara ahora la variable con el valor 64, y vuelve a colocarse otro indicador de BRIGHT (o BR) en 1 ó 0, según corresponda. A continuación se comprueban los otros bits en secuencia, tal y como se muestra en el listado de la figura 6.12.

```
500 REM Decodifica los atributos de color.
510 LET a=ATTR (r,c)
515 REM Comprueba el bit de intermitencia.
520 IF a<128 THEN LET fl=0: GO TO 540
530 LET fl=1: LET a=a-128
535 REM Comprueba el bit de brillo.
540 IF a<64 THEN LET br=0: GO TO 560
550 LET br=1: LET a=a-64
555 REM Comprueba los bits de color de fondo.
560 IF a<32 THEN LET pg=0: GO TO 580.
570 LET pg=1: LET a=a-32
580 IF a<16 THEN LET pr=0: GO TO 600
590 LET pr=1: LET a=a-8
600 IF a<8 THEN LET pb=0: GO TO 620
610 LET pb=1: LET a=a-8
615 REM Comprueba los bits de color de texto.
620 IF a<4 THEN LET ig=0: GO TO 640
630 LET ig=1: LET a=a-4
640 IF a<2 THEN LET ir=0: GO TO 660
650 LET ir=1: LET a=a-2
660 LET ib=a
670 RETURN
```

Fig. 6.12. Subrutina para decodificar atributos de color.

Capítulo 7
Gráficos y diagramas

Con un ordenador, se pueden llevar a cabo muchos cálculos y medidas, y manejar grandes matrices de números. Una vez producidos esos números, un método de presentarlos es producir una lista o quizás una tabla de cifras. Desafortunadamente, esta tabla o lista de números no nos es de gran ayuda para la interpretación de los resultados.

Cuando examinamos una lista de resultados, generalmente estamos más interesados en la forma en que éstos cambian y no en los números concretos.

Un modo mejor de presentar los resultados es mostrarlos visualmente, utilizando un método de visualización de gráficos, o quizás un diagrama. Este tipo de gráficos o diagramas muestra normalmente cada resultado, bien como una línea cuya longitud varía o quizás como un punto cuya altura (sobre una línea de referencia) es proporcional a la cantidad que se va a visualizar. Uno de los tipos de presentación más sencillo consiste en relacionar los valores que se van a considerar con una columna de longitud variable. Un ejemplo de esto último, en la vida real, puede ser el termómetro de mercurio.

Dibujo de un termómetro

Empecemos intentando realizar una presentación tipo termómetro, utilizando para ello los símbolos gráficos de mosaico de que dispone el Spectrum.

En un termómetro de mercurio convencional, la longitud de la columna de mercurio indica la temperatura. Podemos representar la columna de mercurio dibujando una sencilla barra vertical, cuya longitud sea proporcional a la medida que representa. En el caso que nos ocupa, es la temperatura. Podemos dibujar el tubo del termómetro trazando una caja de diferente color alrededor de la columna

de medida. La altura de la caja debe ser suficiente para albergar la columna de medida en su valor máximo.

Para que las lecturas del termómetro tengan sentido, necesitaremos una escala. En los termómetros reales, ésta está dibujada sobre el tubo del termómetro. En nuestra presentación, dibujaremos la escala a lo largo de la columna de medida. Los signos menos se usan como marcas de graduación, midiendo la longitud de la columna. Algunos de ellos también tienen junto a ellos un número, que muestra la correspondiente temperatura en grados Celsius (C). En nuestro caso, sólo se han marcado de este modo las temperaturas superior e inferior. A medida que cambia la temperatura, también cambia la longitud de la columna vertical, y la parte superior de ésta indica la medida de la temperatura.

Suponga que queremos medir desde 0° C hasta 100° C.

Los símbolos de mosaico nos permiten dibujar por pasos de medio espacio de carácter, y por tanto, desde la parte superior de la pantalla hasta la inferior habrá 44 pasos. Una longitud conveniente para la columna puede ser de unas 20 unidades. Por tanto, cada bloque de la columna representa 5 grados C. Con todo esto ya podemos dibujar el tubo del termómetro. El fondo del tubo se crea imprimiendo símbolos de mosaico con códigos 129, 131 y 130, justo en el medio de la fila de texto 20. Utilizaremos un bucle para dibujar el tubo, y para crear las graduaciones imprimiremos símbolos en líneas sucesivas a partir de la línea 20. Finalmente produciremos el final superior del tubo, imprimiendo tres símbolos de mosaico en la línea 8. Los calibres de la escala se escriben en las posiciones apropiadas a lo largo del tubo del termómetro. Para dibujar la columna de mercurio, se escalan las lecturas de la temperatura de 5 en 5 grados, dividiendo t por 5. Observe que antes de realizar la escala añadimos 5 a t. Esto es debido a que el signo menos de los grados C, se encuentra cierta distancia por encima del símbolo inferior en la columna de mercurio.

Después de realizar la escala, los valores de la temperatura se redondean para convertirlos en un número entero, y. A continuación se establece un bucle con un límite de y/2, ya que por cada posición de un símbolo se dan dos pasos. Este bucle imprime espacios de caracteres llenos, comenzando por el fondo del tubo, y llegando a una longitud cercana a los 10 grados. Finalmente se compara y/2 con INT (y/2), para ver si es necesario otro paso más de 5 grados y, si es así, el espacio de carácter siguiente, por arriba, se rellena sólo con la amistad del bloque del símbolo.

En la figura 1, se muestra un programa que produce el dibujo del termómetro en la pantalla de baja resolución.

En la parte superior de la pantalla y en el dibujo del termómetro, se visualizan lecturas aleatorias de la temperatura. En este programa, antes de visualizar cada nueva temperatura, se borra la lectura anterior de la columna de mercurio, imprimiendo bloques compactos en todas las posiciones de la columna, utilizando el comando INVERSE que, efectivamente, vuelve a poner la columna con el color de fondo, o PAPER.

```
100 REM Termometro creado utilizando graficos
de mosaico.
110 CLS
120 INK 0: PAPER 7
130 LET xo=118: LET yo=16
140 REM Dibuja el tubo del termometro.
150 PRINT AT 20,15;CHR$ 129;CHR$ 131;CHR$ 130;
160 FOR n=1 TO 11
170 PRINT AT 20-n,14;"-";CHR$ 133;CHR$ 128;
CHR$ 138;
180 NEXT n
190 PRINT AT 8,15;CHR$ 132;CHR$ 140;CHR$ 136;
200 REM Dibuja la escala.
210 PRINT AT 19,13;"0";
220 PRINT AT 9,11;"100";
230 PRINT AT 14,11;"C";
240 PRINT AT 13,10;"grad";
250 REM Bucle de visualizacion.
260 FOR k=1 TO 100
270 LET t=INT (100*RND)
280 INK 1
290 PRINT AT 1,1;"Temperatura = ";t;
300 PRINT " grados C.     "
310 GO SUB 500
320 PAUSE 200
330 NEXT k
340 STOP
500 INVERSE 1
510 REM Borra la lectura.
520 FOR n=1 TO 11
530 PRINT AT 20-n,16;CHR$ 143;
540 NEXT n
550 INVERSE 0
560 REM Dibuja la nueva lectura.
570 INK 2
580 LET y=INT ((t+5)/5+0.5)
590 FOR n=1 TO INT (y/2)
600 PRINT AT 20-n,16;CHR$ 143
610 NEXT n
620 IF INT (y/2)=y/2 THEN GO TO 650
630 LET y=INT (y/2)
640 PRINT AT 19-y,16;CHR$ 140;
650 RETURN
```

Fig. 7.1. Visualización de un termómetro creado utilizando gráficos de mosaico.

La columna de mercurio se ha dibujado con el color de TEXTO (INK) rojo. El resultado sobre la pantalla es semejante al dibujo que se muestra en la figura 7.2.

Evidentemente, una columna vertical también puede usarse para representar cualquier cantidad que usted quiera, y por tanto, este dibujo puede utilizarse como medidor de fuel, o indicador de velocidad e incluso para indicar los tantos de un juego. Una forma de presentación alternativa puede ser colocar el indicador móvil con la tira en posición horizontal, de forma que parezca un indicador de velocidad típico de coches. En el momento de la elección de la presentación y de su posición en la pantalla, es importante evitar la presencia de dos colores diferentes de tinta en cualquier espacio de símbolo.

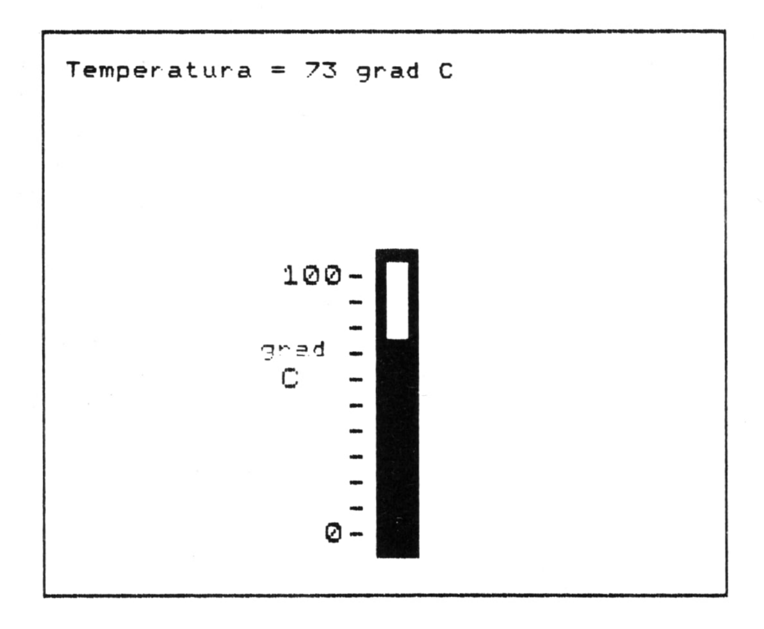


Fig. 7.2. Visualización típica del programa de la figura 7.1.

Un termómetro mejor

El problema principal del dibujo del termómetro, utilizando el modo gráfico de baja resolución, es que los pasos que realiza midiendo la cantidad que deseamos son demasiado grandes. Si cambiamos al modo de alta resolución, podemos obtener lecturas

mucho más precisas. El dibujo del tubo y la columna quizás sea algo más sencillo utilizando la alta resolución, pero, en cuanto al texto que ha de añadirse al dibujo, necesitaremos colocarlo cuidadosamente, teniendo en cuenta las posiciones de los símbolos de texto. Este último punto es importante para evitar problemas con los colores, ya que los gráficos en color están ligados a los espacios de símbolos.

El tubo se dibuja fácilmente, ya que es un rectángulo, y para ello pueden utilizarse los comandos PLOT y DRAW. La producción de marcas a escala no tiene problemas, y utiliza comandos DRAW en un bucle. Por conveniencia, la marca de escala para 0 grados se dibuja separadamente antes del comienzo del bucle de las marcas de la escala. Los valores de la escala y la leyenda "grad C" se imprimen sencillamente en los lugares adecuados, utilizando los comandos PRINT AT.

El dibujo de la columna de mercurio implica la creación de un rectángulo coloreado cuya altura es t unidades, la escala de la temperatura es, en este caso 1:1, y la altura máxima de la columna de mercurio se fija en 100 unidades de pantalla. Con el termómetro de alta resolución, no es necesaria la cuantificación de los grados de 5 en 5, como hacíamos con los gráficos de mosaico, ya que las marcas de graduación de la escala pueden escribirse en cualquier punto de la pantalla. Sin embargo, la posición del tubo necesita elegirse teniendo en cuenta que los símbolos de texto estén en línea con las marcas de la escala. La columna se colorea en ese momento, dibujando seis líneas verticales una al lado de la otra, y todas ellas con t unidades de longitud. Para aprovechar las posibilidades del comando DRAW, se dibujan líneas alternas por encima y por debajo de la posición del cursor, incrementándose x en una unidad después de dibujar cada línea.

En la figura 7.3., se muestra un programa para diseñar un dibujo estilo termómetro, utilizando gráficos de alta resolución. Los resultados sobre la pantalla se muestran en la figura 7.4. Por supuesto, el medidor también podía dibujarse con la barra de medida horizontal. Esto significaría reestructurar la secuencia de dibujo para producir líneas horizontales en lugar de verticales, y reorganizar también los números de la escala y el texto de los rótulos del dibujo, para situarlos en el lugar adecuado, de acuerdo con la nueva columna de medida.

```
100 REM Termometro de alta resolucion.
110 CLS
120 INK 0: PAPER 7
130 LET xo=118: LET yo=16
140 REM Dibuja el tubo del termometro.
150 PLOT xo,yo
160 DRAW 10,0
170 DRAW 0,108
180 DRAW -10,0
190 DRAW 0,-108
200 REM Dibuja la escala.
210 PLOT xo,yo+4
220 DRAW -3,0
230 DRAW 3,0
240 FOR n=1 TO 10
250 DRAW 0,10
260 DRAW -3,0
270 DRAW 3,0
280 NEXT n
290 PRINT AT 19,13;"0";
300 PRINT AT 6,11;"100";
310 PRINT AT 14,12;"C";
320 PRINT AT 13,10;"grad";
330 REM Bucle de visualizacion.
340 FOR k=1 TO 100
350 LET t=INT (100*RND)
360 INK 1
370 PRINT AT 1,1;"Temperatura=";t;
380 PRINT " grados C.    "
390 GO SUB 500
400 PAUSE 200
410 NEXT k
420 STOP
500 INVERSE 1
510 REM Borra la lectura anterior.
520 PLOT xo+2,yo+1
530 LET y=104
540 FOR n=1 TO 6
550 DRAW 0,y: DRAW 1,0
560 LET y=-y
570 NEXT n
580 DRAW 0,y
590 INVERSE 0
600 REM Dibuja el nuevo valor leido
610 INK 2
620 PLOT xo+2,yo+4
630 FOR n=1 TO 6
640 DRAW 0,t: DRAW 1,0
650 LET t=-t
660 NEXT n
670 DRAW 0,t
680 RETURN
```

Fig. 7.3. Visualización de un termómetro con alta resolución.

En este programa, los valores de la temperatura se generan aleatoriamente por el ordenador, y se visualizan también como lecturas de la temperatura en el comienzo de la pantalla. Si utilizamos el

interface entrada/salida adecuado, el Spectrum puede conectarse a un termómetro electrónico. Si éste es el caso, las lecturas de la temperatura pueden introducirse en el Spectrum y luego visualizarse, de forma que la pantalla de presentación actúe como un termómetro real.

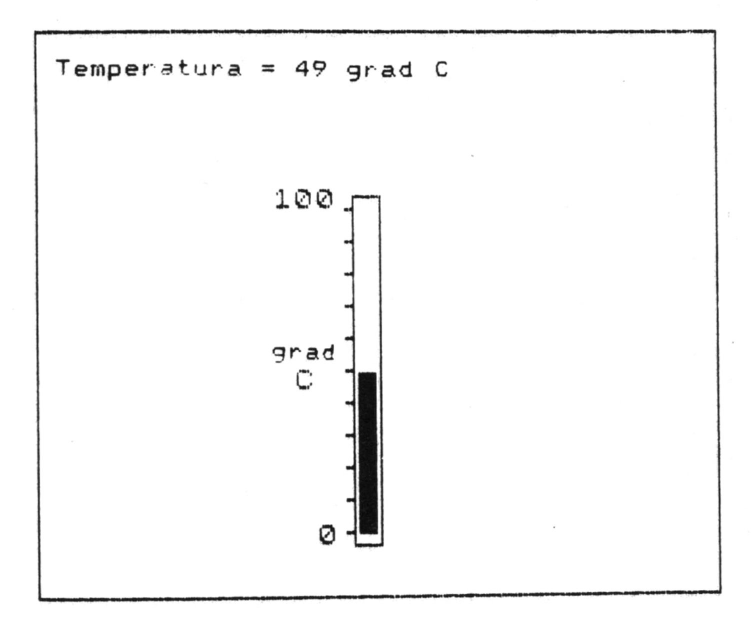


Fig. 7.4. Visualización de un termómetro típico.

Diagrama de barras

El sistema de presentación estilo termómetro puede ser útil para mostrar el estado actual de alguna medida. Sin embargo, es más útil mostrar cómo cambia una situación en un determinado período de tiempo. Podemos, por ejemplo, medir la temperatura de las doce en punto (mediodía) todos los días de la semana. Hecho esto, presentaremos la información fácilmente, dibujando todos los termómetros para todos los días de la semana, uno junto al otro. En esta presentación, la variable longitud de la barra se dibuja para cada día, y también se incluye una escala sencilla en el lado izquierdo. Para mejorar la visibilidad, las barras pueden dibujarse dejando un espacio entre dos barras adyacentes. Este tipo de presentación se llama histograma, pero se conoce comúnmente como diagrama de barras.

Los diagramas de barras no pretenden, normalmente, ser un

dibujo particularmente exacto, ya que su principal aplicación es mostrar la tendencia de la variable que se visualiza. Suelen utilizarse con frecuencia en aplicaciones de negocios, para ver la inclinación de las ventas durante el año, el nivel del stock, número de pedidos o ganancias en un determinado período. Con este tipo de diagramas, es muy sencillo ver la tendencia de los resultados.

Una posible mejora, para este tipo de diagrama de barras, es establecer que el color de la barra cambie, si sobrepasa un nivel determinado por encima o por debajo. Esto proporciona una aviso muy visible de que la situación empieza a ser peligrosa o necesita atención. En estos casos, la barra completa puede cambiar de color, o bien sólo cambiará la parte que sobrepasa el límite.

También pueden usarse los gráficos de mosaico de baja resolución para dibujar diagramas de barras. Aunque la resolución vertical es relativamente tosca, el dibujo resultante para este tipo de diagramas, puede ser muy efectivo.

La figura 7.5., da un listado de un programa que dibuja un diagrama de barras usando gráficos de mosaico. En este programa se dibuja una barra separada para cada día de la semana, y cada barra se traza utilizando la misma técnica que para la columna de mercurio en el programa del termómetro.

```
100 REM Diagrama de barras sencillo
110 REM creado utilizando graficos de mosaico.
120 BORDER 3
130 INK 0: PAPER 7
140 DIM d$(7,2): DIM t(7)
150 REM EStablece los datos.
160 FOR n=1 TO 7
170 READ d$(n),t(n)
180 NEXT n
190 DATA "Lu",60,"Ma",65,"Mi",80
200 DATA "Ju",55,"Vi",65
210 DATA "Sa",70,"Do",65
220 REM Dibuja las escalas.
230 FOR n=1 TO 22
240 PRINT AT 19,7+n;CHR$ 131
250 NEXT n
260 FOR n=1 TO 11
270 PRINT AT 19-n,8;CHR$ 138
280 NEXT n
290 FOR n=1 TO 7
300 PRINT AT 20,7+3*n;d$(n);"  ";
310 NEXT n
320 PRINT AT 18,6;"0";
330 PRINT AT 8,4;"100";
340 PRINT AT 13,5;"F";
350 PRINT AT 12,3;"grad";
360 FOR j=1 TO 7
```

```
370 GO SUB 500
380 NEXT j
390 PRINT AT 2,10;"Temperaturas diarias."
400 STOP
500 INK 2
510 REM Traza la barra.
520 LET y=INT ((t(j)+5)/5+0.5)
530 FOR n=1 TO INT (y/2)
540 PRINT AT 19-n,7+3*j;CHR$ 143;CHR$ 143;
550 NEXT n
560 IF INT (y/2)=y/2 THEN GO TO 590
570 LET y=INT (y/2)
580 PRINT AT 19-y,7+3*j;CHR$ 140;CHR$ 140
590 RETURN
```

Fig. 7.5. Diagrama de barras realizado utilizando gráficos de mosaico.

Los datos de este programa se leen dentro de una matriz, de forma que el trazado de las barras pueda usar un bucle de dibujo común. Es fácil establecer que los datos de la temperatura se introduzcan desde el teclado. Para ello, utilizaremos una instrucción INPUT en lugar de READ.

El dibujo producido en la pantalla es como el que se muestra en la figura 7.6. Si alteramos las escalas y leyendas, este programa puede adaptarse rápidamente a cualquier variable para dibujar el correspondiente diagrama.

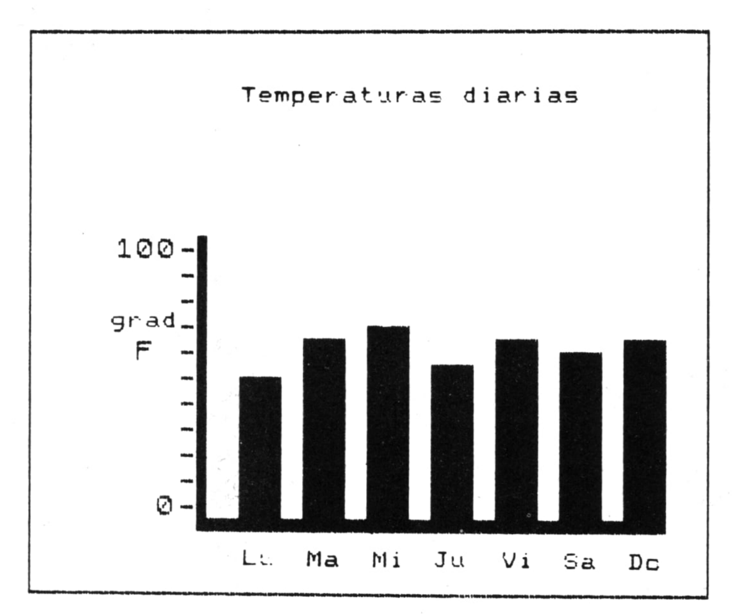


Fig. 7.6. Visualización producida por el programa de la figura 7.5.

Diagrama de barras de alta resolución

La figura 7.7., muestra un programa que dibuja un diagrama de barras utilizando gráficos de alta resolución. La figura 7.8., muestra el resultado en la pantalla. En este programa, las barras se han dibujado de forma diferente de como se había hecho en el dibujo del

```
100 REM  Diagrama de barras de alta resolucion
110 CLS
120 BORDER 3
130 DIM d$(7,2): DIM t(7)
135 REM Establece los datos.
140 FOR n=1 TO 7
150 READ d$(n),t(n)
160 NEXT n
170 DATA "Lu",15,"Ma",18,"Mi",25
180 DATA "Ju",12,"Vi",17,"Sa",20,"Do",18
185 REM Dibuja ejes y escalas.
190 INK 0
200 LET xo=48: LET yo=20
210 PLOT xo,yo
220 DRAW 168,0
230 PLOT xo,yo
240 FOR n=1 TO 6
250 DRAW 0,20
260 DRAW -3,0
270 DRAW 3,0
280 NEXT n
290 PRINT AT 20,7;;
300 FOR j=1 TO 7
310 PRINT d$(j);"  ";
320 NEXT j
330 PRINT AT 19,2;"0";
340 PRINT AT 4,2;"30"
350 PRINT AT 12,2;"C";
360 PRINT AT 11,0;"grad"
365 REM Dibuja barras.
370 INK 2
380 PLOT xo,yo
390 DRAW 4,0
400 FOR k=1 TO 7
410 DRAW 8,0
420 LET y=t(k)*4
430 FOR n=1 TO 4
440 DRAW 0,y
450 DRAW 1,0
460 DRAW 0,-y
470 DRAW 1,0
480 NEXT n
490 DRAW 8,0
500 NEXT k
510 REM Escribe la leyenda.
520 INK 1
530 PRINT AT 2,6;"Temperatura de cada dia";
540 STOP
```

Fig. 7.7. Programa de creación de un diagrama de barras de alta resolución.

termómetro. En este caso, el límite del bucle se fijó en la lectura deseada, medida en unidades de pantalla, y se dibujó una serie de líneas cortas horizontales, una sobre la otra, para producir una barra coloreada. Esta técnica necesita más pasos por el bucle que la versión de las líneas verticales, pero es igualmente eficaz en la producción de barras. Como en el caso del termómetro, la posición de las barras debe elegirse con mucho cuidado en relación con las posiciones de los símbolos de texto, para evitar problemas con los colores que se van a visualizar.

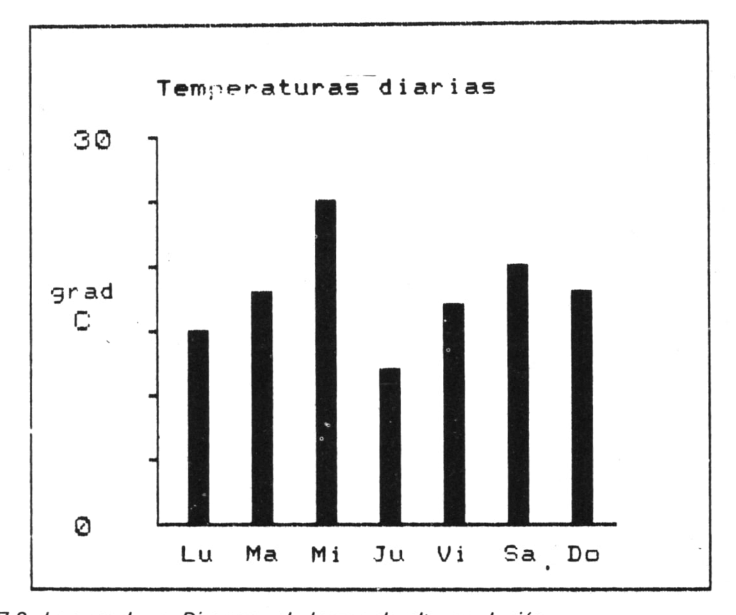


Fig. 7.8. Imagen de un Diagrama de barras de alta resolución.

Diagrama de barras múltiple

Para visualizar dos variables diferentes en el mismo diagrama, las barras se dibujan a pares, de forma que resulten intercaladas. Para que los dos conjuntos de barras sean claramente diferenciables, deben elegirse colores diferentes para cada conjunto. Pueden intercalarse casi tres o quizás cuatro gráficos. Algunas barras pueden dibujarse como cajas abiertas, bordeadas por otros colores diferentes. Una aplicación típica de un diagrama de barras múltiple puede ser la presentación de las ventas y las ganancias en un mismo diagrama. Puede resultar muy útil, quizás, para mostrar y predecir las tendencias.

El programa listado en la figura 7.9., es un ejemplo de diagrama de barras múltiples, que utiliza alta resolución, y produce un diseño de temperaturas máximas y mínimas para todos los días de la semana. En este caso, uno de los conjuntos de barras se ha dibujado en rojo, mientras que el otro se ha dibujado en azul claro.

```
100 REM diagrama de barras multiple.
110 CLS
120 BORDER 3
130 DIM d$(7,2)
140 DIM l(7)
150 DIM h(7)
160 REM Establece los datos.
170 FOR n=1 TO 7
180 READ d$(n),l(n),h(n)
190 NEXT n
200 DATA "Lu",7,15,"Ma",10,18,"Mi"
210 DATA 15,25,"Ju",5,12,"Vi",2,17
220 DATA "Sa",7,20,"DO",15,18
230 REM Dibuja los ejes y las escalas.
240 INK 0
250 LET xo=48: LET yo=20
260 PLOT xo,yo
270 DRAW 168,0
280 PLOT xo,yo
290 FOR n=1 TO 6
300 DRAW 0,20
310 DRAW -3,0
320 DRAW 3,0
330 NEXT n
340 PRINT AT 20,7;;
350 FOR j=1 TO 7
360 PRINT d$(j);"  ";
370 NEXT j
380 PRINT AT 19,4;"0";
390 PRINT AT 4,3;"30";
400 PRINT AT 12,2;"c";
410 PRINT AT 11,1;"grad"
420 REM Traza las barras.
430 PLOT xo+8,yo
440 FOR k=1 TO 7
450 INK 5
460 LET y=l(k)*4
470 FOR n=1 TO 4
480 DRAW 0,y
490 DRAW 1,0
500 DRAW 0,-y
510 DRAW 1,0
520 NEXT n
530 DRAW 4,0
540 INK 2
550 LET y=h(k)*4
560 FOR n=1 TO 4
570 DRAW 0,y
580 DRAW 1,0
590 DRAW 0,-y
```

```
600 DRAW 1,0
610 NEXT n
620 DRAW 4,0
630 NEXT k
640 REM Imprime la leyenda.
650 INK 1
660 PRINT AT 1,3;"Temperaturas para cada dia";
670 INK 5
680 PRINT AT 3,6;"Min    ";CHR$ 143;
690 PRINT CHR$ 143;CHR$ 143;
700 INK 2
710 PRINT AT 3,17;"Max    ";CHR$ 143;
720 PRINT CHR$ 143;CHR$ 143;
730 STOP
```

Fig. 7.9. Programa para la creación de un diagrama de barras múltiple.

En este tipo de diagrama se debe incluir siempre una leyenda para mostrar lo que representa cada conjunto de barras. La figura 7.10., muestra el tipo de visualización que produce este programa.

Los gráficos de barras se usan frecuentemente en diagramas de temas financieros y de producción, ya que es mucho más agradable y sencillo presentar un diagrama de barras que presentar una lista de valores.

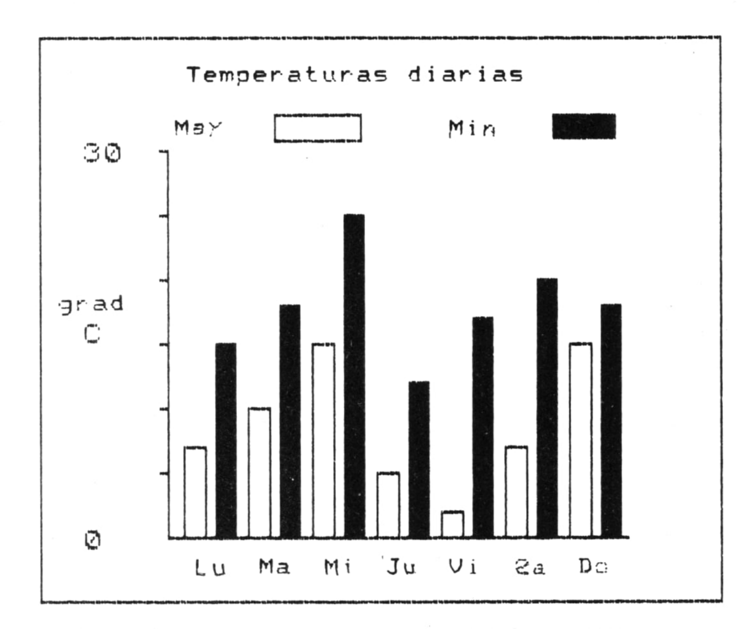


Fig. 7.10. Aspecto típico que presenta un diagrama de barras múltiple.

Gráficos científicos

Aunque los diagramas de barras son muy adecuados para los negocios, en el campo científico o matemático el trazado de gráficos exige un diseño diferente, ya que debe presentar los resultados con más exactitud.

La presentación es similar a la del gráfico de barras, con los resultados de los cálculos o experimentos trazados verticalmente en la pantalla y las medidas tomadas horizontalmente. En este caso, sin embargo, el valor de Y se muestra sencillamente como un punto situado en un lugar equivalente al de la parte superior de la barra en el diagrama de barras. Algunas veces, para hacer más visible el punto, puede utilizarse en su lugar un pequeño signo más, un triángulo o un círculo.

En el diagrama de barras, las variables normalmente son positivas, pero en gráficos científicos, las variables X o Y pueden ser positivas o negativas. Para poder albergar estos valores, los ejes X e Y se dibujan tal y como se muestra en la figura 7.11.

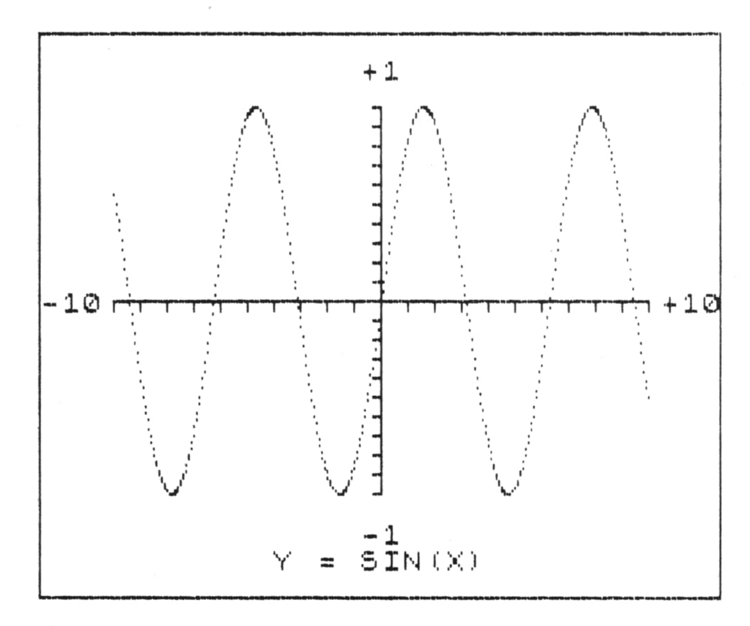


Fig. 7.11. Gráfica de una función seno.

Los valores positivos de X se dibujan a la derecha del eje Y vertical, y los negativos a la izquierda. Del mismo modo, los valores positivos de Y se escriben por encima del eje X, y los valores negati-

vos por debajo. Si no existen valores negativos de X, no se dibujará la mitad izquierda del gráfico, y el eje Y aparecerá en la parte izquierda del diagrama. Del mismo modo, si no hay valores negativos de X, sólo se dibujará la parte superior del gráfico, sobre el eje X. Algunas veces sólo necesitaremos dibujar la cuarta parte del gráfico completo, para visualizar todos los puntos necesarios. La ventaja de dibujar sólo una parte del sistema de ejes X, Y, es que la parte elegida puede aumentarse y llenar la pantalla, obteniendo con ello una mejor resolución.

Para ver cómo se produce este tipo de gráficos, dibujemos el de la ecuación Y = SIN(X), con valores para X desde —10 hasta 10. Los valores que tomará SIN(X) irán desde —1 hasta +1, y estos valores son los que determinan la escala del eje Y. Para producir un gráfico de un tamaño razonable, debemos multiplicar Y por una escala. Una que nos puede convenir puede ser 60 divisiones de pantalla por cada unidad de Y, de modo que simplemente multiplicaremos los valores de Y por un factor de escala ys, antes de calcular las coordenadas del dibujo. El valor de ys se fija en 60 al comienzo del programa. Del mismo modo, se usa un multiplicador (escala) xs para los valores de X. En este caso, xs se fija en 10, al comienzo del programa, para dar un gráfico de una anchura de 200 unidades de pantalla. Los valores de xs e ys pueden elegirse de acuerdo con la gama de valores que tomarán en el dibujo x e y, para obtener el mayor gráfico que quepa en la pantalla.

El primer paso en la construcción del gráfico es producir las líneas de los ejes X e Y, y las escalas. Esto puede hacerse con facilidad usando los comandos DRAW y dos sencillos bucles de dibujo. En primer lugar colocamos el cursor gráfico en el centro de los ejes del gráfico, utilizando PLOT 128,88, que coloca un punto en el centro de la pantalla. El paso siguiente es dibujar la parte derecha del eje X, que se obtiene dibujando una serie de líneas horizontales cortas seguidas por una línea vertical corta que va debajo de la línea del eje y otra segunda línea vertical para traer el cursor al eje. La longitud de cada paso es igual al incremento de x multiplicado por xs. Después de dibujar la parte derecha, se repite el bucle y las líneas se dibujan a la izquierda. El eje Y y sus marcas de escala se dibujan de forma semejante. El dibujo completo de los ejes se ejecuta mediante una subrutina, aunque puede hacerse en el programa principal, si lo deseamos. La última parte de la subrutina imprime las marcas de la escala en la parte final de dos ejes.

Después del dibujo de los ejes, el siguiente paso es el trazado del propio gráfico. En este caso, los cálculos se llevan a cabo en un bucle

con el ángulo (X) aumentando en pequeños incrementos desde —10 hasta +10. La coordenada x (xp), para cada punto, se calcula del siguiente modo:

xp = xo + (xs*x)

donde xo es el valor de x para el centro del gráfico que, en este caso, es 128.

Como utilizamos un factor de escala, xs, podemos, si lo deseamos, alterar fácilmente el tamaño del gráfico. También podemos alterar el valor de xo, y con ello, colocaremos el gráfico en cualquier posición sobre la pantalla.

Para trazar los puntos en el gráfico, calcularemos el valor de Y, para el comando PLOT, del siguiente modo:

yp = yo + ys * SIN(x)

y se traza el punto con la siguiente expresión:

PLOT xp,yp

El listado del programa se puede ver en la figura 7.12., y el resultado que se produce en la pantalla es similar al de la figura 7.11. En este programa, los colores INK Y PAPER son sencillamente blanco y negro, pero pueden obtenerse gráficos en color cambiando los colores de texto (INK) y de fondo (PAPER) a cualquier combinación.

```
100 REM Representacion grafica del seno.
110 BORDER 5
120 CLS
130 LET xs=10
140 LET ys=60
150 LET xo=128
160 LET yo=92
170 REM Dibuja los ejes.
180 GO SUB 500
190 REM Traza el dibujo.
200 FOR x=-10 TO 10 STEP 0.1
210 LET y=SIN x
220 LET xp=xo+INT (xs*x)
230 LET yp=yo+INT (ys*y)
240 PLOT xp,yp
250 NEXT x
260 REM Escribe la leyenda
270 PRINT AT 20,11;"Y=SIN (X)";
280 STOP
498 REM
```

```
499 REM Subrutina de dibujo del eje.
500 LET x=xs
510 REM Traza el eje X.
520 FOR k=1 TO 2
530 PLOT xo,yo
540 FOR j=1 TO 10
550 DRAW x,0
560 DRAW 0,-3
570 DRAW 0,3
580 NEXT j
590 LET x=-x
600 NEXT k
610 LET y=ys/10
620 REM Traza el eje Y
630 FOR k=1 TO 2
640 PLOT xo,yo
650 FOR j=1 TO 10
660 DRAW 0,y
670 DRAW -3,0
680 DRAW 3,0
690 NEXT j
700 LET y=-y
710 NEXT k
720 PRINT AT 10,0;"-10";
730 PRINT AT 10,29;"+10";
740 PRINT AT 1,15;"+1";
750 PRINT AT 19,15;"-1";
760 RETURN
```

Fig. 7.12. Programa que dibuja gráficos de la función seno.

También es posible dibujar dos o más gráficos sobre los mismos ejes, utilizando diferentes colores de texto (INK) para los puntos del segundo gráfico. Un problema aquí, es que, en ciertos lugares, los puntos del primer gráfico ocupan el mismo espacio de carácter que los puntos del segundo gráfico, y su color cambiará al del segundo gráfico. En la mayoría de los casos esto no resulta una gran dificultad, a no ser que las líneas de las dos curvas transcurran muy próximas una a otra.

Unión de los puntos

Para obtener un buen dibujo de la curva producida por la función seno, deberemos trazar muchos puntos, de forma que haya muy poco espacio entre ellos. Si tomáramos menos puntos para x e y, los puntos tenderían a separarse dando una impresión mucho menos clara de la forma de la función.

Algunas veces, podemos desear encontrar el valor probable para la y para un valor de x no incluido en los puntos que se utilizaron

para el gráfico. Podemos obtener este valor aproximado de un punto de la curva mediante una técnica llamada interpolación.

La técnica más sencilla de interpolación consiste en unir puntos sucesivos de la curva con líneas rectas. A este tipo de interpolación se le denomina interpolación lineal. Podemos, de hecho, unir los puntos con líneas rectas mientras realizamos el gráfico. Este método nos proporciona una curva más fácil de seguir cuando el número disponible de puntos es limitado. Se necesita, sin embargo, poner mucho cuidado, ya que, si se utilizan muy pocos puntos, la técnica de interpolación puede ser muy imprecisa.

Para unir los puntos, la rutina de trazado de puntos se altera y, en lugar de utilizar un comando PLOT para trazar cada punto, se utiliza un comando DRAW para dibujar una línea corta desde el último punto dibujado hasta el nuevo punto. Se necesitará un par de nuevas variables, x1 e y1 para especificar el último punto trazado. Las variables x2 e y2 se usan para cada nuevo punto. Después de dibujar cada línea, x1 e y1 se actualizan y toman valores iguales a las coordenadas (x2, y2) del último punto del gráfico. Los valores x, y, para el comando DRAW se calculan sencillamente tomando la diferencia entre los valores de x2, y2 y x1, y1. El Spectrum moverá automáticamente su Cursor gráfico al nuevo punto mientras va dibujando la línea. El primer punto debe trazarse utilizando un comando PLOT, con el fin de colocar el cursor gráfico en la posición adecuada en el gráfico. Esto puede hacerse calculando un valor inicial de x1 e y1 para el primer punto que debe dibujarse. En la figura 7.13, se muestra una variación del programa sencillo de trazado de gráficos, que utiliza interpolación para la unión de los puntos. En este programa se dibuja la curva de un coseno, cuyo aspecto es semejante a la curva de un seno, pero con su posición desplazada sobre el eje x, como puede verse en la figura 7.14.

```
100 REM Grafica del coseno realizada
encadenando puntos.
110 BORDER 2
120 CLS
130 LET xs=10
140 LET ys=60
150 LET xo=128
160 LET yo=92
170 REM Dibuja los ejes.
180 GO SUB 500
190 REM Traza el dibujo.
200 LET x1=xo+xs*-10
210 LET y1=yo+INT (ys*COS -10)
220 PLOT x1,y1
```

```
230 FOR x=-10 TO 10 STEP 0.3
240 LET x2=xo+INT (xs*x)
250 LET y2=yo+INT (ys*COS x)
260 DRAW x2-x1,y2-y1
270 LET x1=x2
280 LET y1=y2
290 NEXT x
300 REM Escribe la leyenda.
310 PRINT AT 20,12;"Y = COS (X)";
320 STOP
498 REM
499 REM Subrutina de dibujo deleje.
500 LET x=xs
510 REM Traza el eje X.
520 FOR k=1 TO 2
530 PLOT xo,yo
540 FOR j=1 TO 10
550 DRAW x,0
560 DRAW 0,-3
570 DRAW 0,3
580 NEXT j
590 LET x=-x
600 NEXT k
610 LET y=ys/10
620 REM Traza el eje Y.
630 FOR k=1 TO 2
640 PLOT xo,yo
650 FOR j=1 TO 10
660 DRAW 0,y
670 DRAW -3,0
680 DRAW 3,0
690 NEXT j
700 LET y=-y
710 NEXT k
720 PRINT AT 10,0;"-10";
730 PRINT AT 10,29;"+10";
740 PRINT AT 1,15;"+1";
750 PRINT AT 19,15;"-1";
760 RETURN
```

Fig. 7.13. Programa para dibujar la función COS por interpolación.

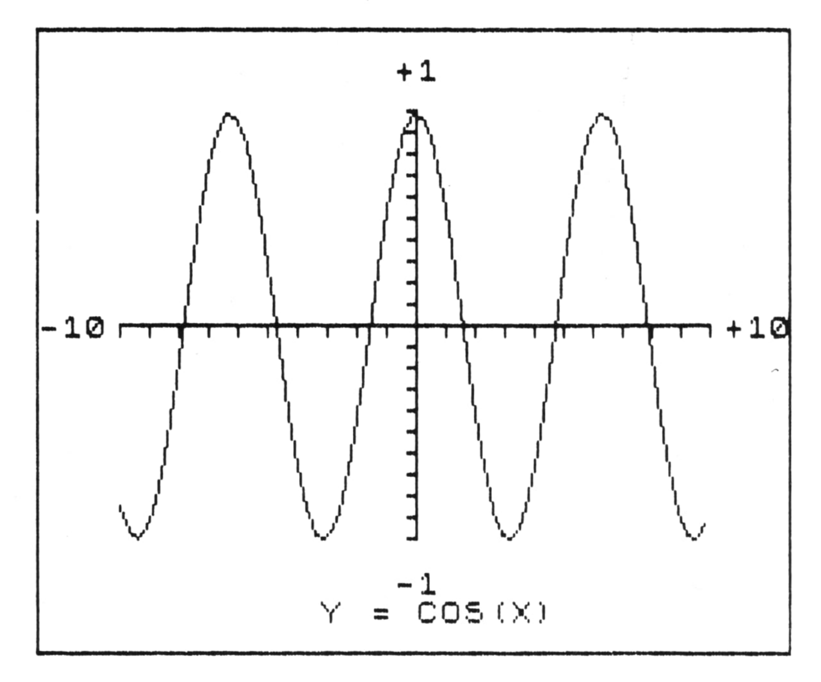


Fig. 7.14. Gráfico producido por el programa COS.

Diseños tipo dial y reloj

En algunas aplicaciones puede necesitarse un diseño tipo dial con indicador, o uno tipo reloj. Un uso típico puede ser un panel de instrumentos de un programa simulador de vuelo, o quizás un instrumento de lectura para un experimento donde el ordenador vigile los resultados. Evidentemente, en algunos casos, el diagrama puede mostrar el tiempo que queda o el que se ha consumido en un programa de juegos.

La presentación básica consiste en un círculo o quizás un polígono, con un dial y uno o dos indicadores. Los indicadores pueden ir radialmente desde el centro del dial. El dial mismo, también puede colorearse para resaltar del fondo. Alrededor del dial y por el exterior, se dibuja algún tipo de escala .

Cuando las lecturas no necesiten ser muy precisas, y se quiera ver la tendencia inmediatamente, este diseño es el adecuado. Un ejemplo de ello son los relojes de todo tipo, digitales y analógicos. Aunque la visualización digital es mucho más precisa, es mucho más sencillo ver la hora con una mirada rápida a una esfera de reloj convencional.

El trazado del dial es muy simple, ya que implica el dibujo de un círculo que puede realizarse utilizando el comando CIRCLE. La escala es simplemente un círculo más grande. Para dibujar las graduaciones de la escala, elegiremos un valor para el radio. Así pues, haremos s1 el radio del final interior de las marcas de la escala, y s2 el radio exterior del final de las marcas. La posición de cada marca se calcula utilizando ecuaciones de rotación, y se dibuja una línea corta radialmente hacia fuera del círculo de la escala. También se usa otro nuevo radio para localizar la posición de los símbolos de texto.

Un punto importante a notar sobre el indicador es que las graduaciones aumentan en el sentido del giro de las agujas del reloj. Sin embargo, en las ecuaciones de rotación normales ocurrirá lo contrario. La posición del indicador se calcula rápidamente utilizando ecuaciones de rotación modificadas. El ángulo de rotación necesario será la relación entre la lectura de la escala y la escala completa, multiplicado por el ángulo total (que corresponde a la escala completa). Si se utilizan 360 grados, el ángulo TH vendrá dado por:

TH = 2*PI*X/FS

donde X es el valor medido, FS es la lectura de la escala completa y TH es el ángulo de rotación. Para invertir la dirección normal de giro, se invierte el signo del termino y.

Algunas veces, el dial puede cubrir sólo 90, 180 ó 270 grados. En este caso, el término 2*PI de la ecuación anterior debe reducirse al valor elegido como ángulo total de la escala, medido en radianes. El valor del ángulo en radianes se encuentra fácilmente utilizando la siguiente ecuación:

RAD = GRAD * PI / 180

Normalmente las ecuaciones de rotación suponen que el punto cero está en la horizontal y a la derecha. Si usted desea que el punto cero se encuentre en la parte superior, como en una brújula (ej., norte = 0), se añadirán 90 grados o PI/2 a los valores de TH, antes de calcular los valores de x e y. Observe que en este programa lo conseguimos utilizando la función SEN para el cálculo de la x, y la función COS para el cálculo de la y, lo cual produce el mismo efecto que girar 90 grados y cambiar el signo de la y.

El dibujo de las marcas de la escala es muy similar al dibujo del indicador, excepto que el comienzo de la línea de la marca se encuentra en un radio más grande que el dial del círculo. Los finales

interior y exterior de la marca se calculan utilizando la ecuación de rotación con dos valores diferentes para la r. El centro de cualquier texto que se utilice para calibrar la escala puede calcularse del mismo modo utilizando un radio mayor que el otro radio de las marcas de la escala. Recuerde que, como hemos encontrado ya el punto centro del texto, debemos escribir cada símbolo utilizando el comando DRAW y el tipo de cadena apropiado.

Normalmente el indicador se vuelve a dibujar para cada nueva lectura, y el indicador antiguo se borra, volviendo a rellenar su lugar con el color del dial, o el color de fondo, si no se coloreó. Puede utilizarse un procedimiento para borrar y reescribir el indicador cada vez que se realiza una nueva lectura.

El programa que se lista en la figura 7.15., produce un dial sencillo con un único indicador, y crea un dibujo similar al que se muestra en la figura 7.16.

En este programa, el dial comienza con el indicador apuntando hacia arriba, y tiene una escala de 270 grados.

```
100 REM Visualizacion del marcador movible.
110 LET xc=128
120 LET yc=88
130 LET r=40
140 LET s1=r+5
150 LET s2=r+10
160 LET st=r+20
170 REM Dibuja el dial.
180 CIRCLE xc,yc,r
190 REM Dibuja la escala.
200 LET dt=1.5*PI/s1
210 LET th=0
220 LET x1=0
230 LET y1=s1
240 PLOT xc+x1,yc+y1
250 FOR n=1 TO s1
260 LET th=th+dt
270 LET x2=s1*SIN th
280 LET y2=s1*COS th
290 DRAW x2-x1,y2-y1
300 LET x1=x2
310 LET y1=y2
320 NEXT n
330 LET dt=1.5*PI/6
340 LET th=0
400 FOR n=0 TO 6
410 LET th=n*dt
420 LET x1=s1*SIN th
430 LET x2=s2*SIN th
440 LET y1=s1*COS th
450 LET y2=s2*COS th
460 PLOT xc+x1,yc+y1
470 DRAW x2-x1,y2-y1
```

```
480 LET xt=xc+st*SIN th
490 LET yt=yc+st*COS th
500 GO SUB 1000
510 NEXT n
520 REM Visualiza el indicador.
530 LET p1=8
540 LET fs=6
550 LET rp=r-5
560 FOR k=1 TO 20
570 FOR p=0 TO 6 STEP 0.2
580 GO SUB 700
590 NEXT p
600 FOR p=6 TO 0 STEP -0.2
610 GO SUB 700
620 NEXT p
630 NEXT k
640 STOP
690 REM Subrutina del indicador.
700 LET th=1.5*PI*p1/fs
710 REM Borra la ultima lectura.
720 INVERSE 1
730 PLOT xc,yc
740 DRAW rp*SIN th,rp*COS th
750 INVERSE 0
760 LET th=1.5*PI*p/fs
770 REM Dibuja el nuevo indicador.
780 LET th=1.5*PI*p/fs
790 PLOT xc,yc
800 DRAW rp*SIN th,rp*COS th
810 LET p1=p
820 RETURN
830 REM Subrutina de simbolos de texto.
1000 PRINT AT 0,0;STR$ (n);
1010 REM Copia el simbolo.
1020 FOR j=0 TO 7
1030 FOR i=0 TO 7
1040 IF POINT (i,175-j)=0 THEN GO TO 1060
1050 PLOT xt+i-4,yt-i+4
1060 NEXT i
1070 NEXT j
1080 PRINT AT 0,0;" ";
1090 RETURN
```

Fig. 7.15. Programa que visualiza un dial.

Si son necesarias dos agujas, como en un dibujo de un reloj convencional, puede utilizarse la misma rutina básica, creando radios diferentes para los indicadores. Del mismo modo, también puede añadirse un tercer indicador. Si los punteros tienen formas diferentes, puede convenirnos separar los procedimientos de dibujo para cada indicador.

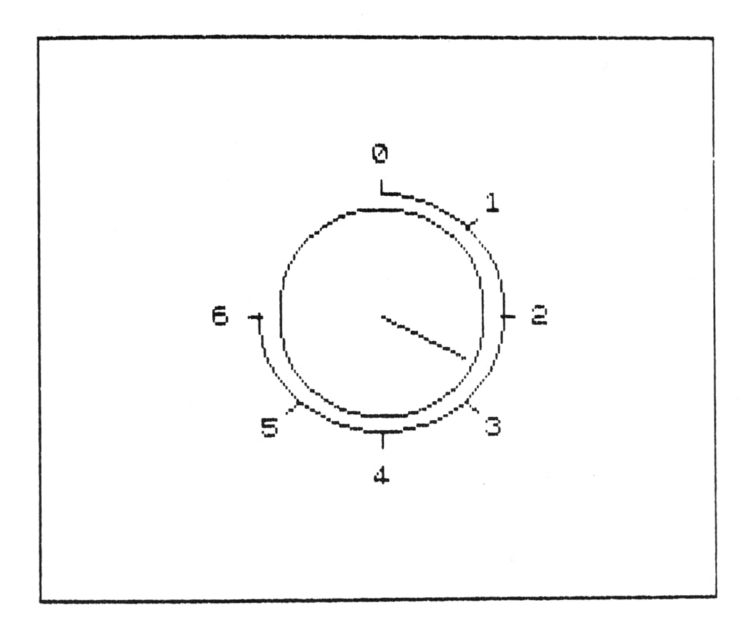


Fig. 7.16. Dibujo típico en la pantalla de indicador de aguja.

Diagramas circulares (de queso).

Una forma de presentación de diagramas, muy atractiva y utilizada con frecuencia en documentos empresariales, es el diagrama circular. Este tipo de diagrama muestra las proporciones en las que se divide el total de un elemento. Un ejemplo de ello puede ser el porcentaje de votos que ha obtenido un partido político en una votación de una muestra de electores. Hemos visto muchas veces este tipo de diagramas en la televisión. Otra aplicación puede ser el mostrar cómo se utilizan los recursos de una compañía, o cómo se gasta el dinero de ésta.

Como su nombre indica, el diagrama de queso es, efectivamente, como una perspectiva desde la parte superior de un queso que se parte en pedazos de varios tamaños. Cada pedazo del queso representa un elemento y muestra el porcentaje del total que supone este elemento. En la figura 7.17., se muestra un diagrama de queso típico.

Para dibujar un diagrama de queso, debemos trazar una serie de segmentos circulares. El ángulo para cada segmento puede calcularse como un porcentaje de 2 x PI (360 grados). Con el Spectrum, la técnica básica consiste en comenzar dibujando el círculo exterior,

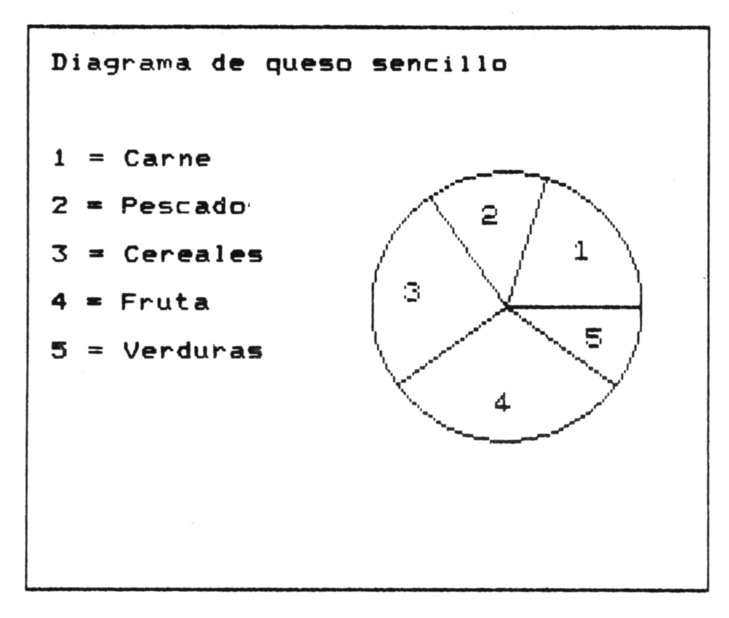


Fig. 7.17. Visualización típica de un diagrama circular o de queso.

utilizando simplemente el comando CIRCLE. Una vez dibujado el círculo, el siguiente paso es dibujar unas series de líneas radiales que separan los pedazos del queso. Las coordenadas de los extremos de esas líneas pueden calcularse por trigonometría, como sigue:

$$XR = DX * COS\ (TH) - DY * SIN\ (TH)$$
$$XR = DX * SIN\ (TH) + DY * COS\ (TH)$$
$$TH = P * 2 * PI/1\varnothing\varnothing$$

donde P es el porcentaje del queso completo representado por el segmento que se está dibujando. DX y DY son las coordenadas equivalentes para la última línea radial trazada. Se calcula a continuación una nueva pareja de valores XR e YR para permitir el dibujo de la línea que separa el segmento. Después de dibujar el segmento, los valores de DX y DY se actualizan, haciéndolos iguales a XR e YR respectivamente y dejando ya todo preparado para dibujar el siguiente sector.

En la figura 7.18., se muestra un sencillo programa que dibuja un diagrama de queso con posibilidad de dibujar hasta cinco segmentos. En este caso, no puede insertarse fácilmente el número del segmento o cualquier otra identificación utilizando el comando PRINT, porque es evidente que la posición necesaria para el sím-

bolo de texto no será una posición normal de colocación de un símbolo de texto. Para superar este problema, el número de identificación para el sector se coloca en su posición, copiándolo a punto con la técnica descrita en el Capítulo Cinco.

```
100 REM Diagrama de queso.
110 DIM s(5)
120 REM Establece los datos.
130 FOR n=1 TO 5
140 READ s(n)
150 NEXT n
160 DATA 20,15,25,30,10
170 REM Dibuja un circulo.
180 LET xc=180
190 LET yc=88
200 LET r=50
210 CIRCLE xc,yc,r
220 REM Marca los sectores.
230 LET dx=r
240 LET dy=0
250 LET th=0
260 FOR n=1 TO 5
270 LET xr=dx*COS th-dy*SIN th
280 LET yr=dx*SIN th+dy*COS th
290 REM  Dibuja la linea de un sector.
300 PLOT xc,yc
310 DRAW xr,yr
320 LET dx=xr
330 LET dy=yr
340 REM Escribe el numero en un sector.
350 LET th=PI*s(n)/100
360 LET xr=dx*COS th-dy*SIN th
370 LET yr=dx*SIN th+dy*COS th
380 LET xt=xc+INT (.7*xr)
390 LET yt=yc+INT (.7*yr)
400 LET a$=STR$ (n)
410 GO SUB 600
420 LET dx=xr
430 LET dy=yr
440 NEXT n
450 REM Escrbe la leyenda.
460 PRINT AT 1,1;" Diagrama de queso";
470 PRINT AT 4,1;"1 = Carne";
480 PRINT AT 6,1;"2 = Pescado";
490 PRINT AT 8,1;"3 = Cereales";
500 PRINT AT 10,1;"4 = Fruta";
510 PRINT AT 12,1;"5 = Verduras";
520 STOP
600 PRINT AT 0,0;a$;
610 FOR j=0 TO 7
620 FOR i=0 TO 7
630 IF POINT (i,175-j)=0 THEN GO TO 650
640 PLOT xt+i-4,yt-j+4
650 NEXT i
660 NEXT j
670 PRINT AT 0,0;" ";
680 RETURN
```

Fig. 7.18. Programa que dibuja un diagrama de queso.

El símbolo que necesitamos se imprime inicialmente en la posición 0,0, y nos suministra un patrón de puntos para el copiado. Las coordenadas para el carácter se calculan (para el ángulo superior izquierdo) añadiendo un pequeño desplazamiento a las coordenadas del punto central del segmento, donde va a colocarse el símbolo. La posición del número correspondiente se calcula realizando la rotación necesaria para dibujar el sector en dos partes. En primer lugar se hace una rotación con un ángulo de medio sector, luego se dibuja el texto, y finalmente, se lleva a cabo el resto de la rotación. Con este sistema el símbolo de texto se coloca en la mitad del sector.

Cuando se utilizan números o letras para identificar los sectores, debe incluirse algún tipo de clave que indique lo que representa cada sector en el diagrama. Este texto, evidentemente, puede escribirse utilizando PRINT o quizás PRINT AT.

Capítulo 8
El mundo en movimiento

La mayoría de los juegos con ordenador necesita producir objetos moviéndose por la pantalla de visualización. Como vimos en el Capítulo Uno, la pantalla de televisión presenta venticinco imágenes por segundo, y, dada la respuesta de nuestros ojos, no vemos las fluctuaciones de las imágenes mientras éstas se trazan. Si establecemos que un objeto se mueva a una posición algo diferente en cada pasada por la pantalla, al espectador le parecerá que se está moviendo por ésta. Este es el principio básico utilizado en la producción de películas de dibujos animados. Para obtener un movimiento de una suavidad aceptable, los cambios de las imágenes se deben realizar al menos diez veces por segundo. Si el cambio en el movimiento es pequeño, el movimiento parecerá suave, pero, con movimientos mayores entre dos imágenes sucesivas, el movimiento será a sacudidas.

En la forma más sencilla de animación, los objetos, como por ejemplo una pelota, un invasor o una nave espacial, se mueven por la pantalla de una posición a otra. Para que los resultados sean más reales, podemos necesitar que cambie la forma del objeto de la pantalla, según se vaya moviendo. Un ejemplo de esto puede ser un hombre que anda por la pantalla. Si la imagen del hombre fuera constante, parecería que se resbalaba por la pantalla como un patinador sobre hielo. Para dar la impresión de marcha, la posición de las piernas e incluso la de los brazos del hombre, deben cambiar según se mueve la imagen de una posición a otra en la pantalla. En efecto, presentaremos una serie de imágenes casi iguales y en rápida sucesión. Muchas acciones, como, por ejemplo, la marcha, son repetitivas, y, por tanto, puede que necesitemos tres o cuatro imágenes diferentes, y repetir la secuencia del movimiento del hombre por la pantalla. En el caso de un pájaro que vuela en la pantalla, las alas se moverán, pero en la vida real el movimiento de las alas no es el sencillo de arriba y abajo que podemos imaginar. De hecho, la

forma completa del pájaro cambia cada vez que aletea en el aire, y, para que los resultados sean reales, tendremos que reproducir los mismos cambios en su forma.

Sencillo movimiento de una pelota

En muchos programas de juegos, lo que se mueve por la pantalla es una simple pelota. Una posibilidad puede ser utilizar un símbolo de texto, por ejemplo un asterisco, para la pelota, y luego imprimirlo en otra nueva posición de carácter alineada con su posición actual. Para evitar ir dejando rastros de símbolos por la pantalla, necesitaremos borrar el símbolo de la posición anterior. Esto puede realizarse fácilmente imprimiendo un símbolo de espacio sobre él.

Comencemos observando el proceso que se lleva a cabo cuando se visualiza en la pantalla una sencilla pelota, que nosotros representamos utilizando el símbolo gráfico de un bloque lleno.

Este símbolo tiene un código de carácter 143. La posición de la pelota puede fijarse utilizando dos variables x e y para dar a la pelota las coordenadas de posición de pantalla. Podemos comenzar colocando la pelota en el centro de la pantalla con x = 15 e y = 1Ø, utilizando:

```
190 PRINT AT y,x;CHR$ 143;
```

El movimiento horizontal de la pelota por la pantalla puede ser muy sencillo, ya que lo único que tenemos que hacer es añadir una unidad al valor de x para mover la pelota a la derecha, o restarlo de x, si se desea mover la pelota a la izquierda. Así pues, tenemos un nuevo valor x, que llamaremos xn, y es el que va a utilizarse en la instrucción PRINT AT para imprimir el símbolo de la pelota en su nueva posición. Usted puede decir que por qué no hemos alterado el valor de x. Una razón es que necesitamos todavía a x para que nos permita imprimir un espacio en blanco sobre el símbolo previo, para borrarlo. Antes de realizar el siguiente movimiento, se fija el valor de x igual a xn, y ya estamos preparados para llevar a cabo de nuevo el proceso completo para el siguiente movimiento de la pelota.

En cuanto al movimiento vertical, necesitamos cambiar la coordenada y. Si añadimos una unidad a y, moveremos la pelota hacia la parte inferior de la pantalla, y, si sustraemos una unidad, nos moveremos hacia la parte superior. Una vez más podemos usar una segunda variable yn para la nueva posición. En lo que se refiere al

movimiento horizontal, podemos imprimir la pelota en su nueva posición y luego borrar el símbolo de la posición anterior, sobreimprimiéndole un espacio. Después del borrado, la variable y se fija como yn, y todo queda listo para el siguiente paso.

El movimiento en diagonal es un proceso algo más complicado, ya que necesitamos alterar los dos valores, x e y en cada paso. Si queremos mover el punto hacia arriba y hacia la derecha, los nuevos valores de xn e yn serán x+1 e y—1 respectivamente. Hacia arriba y hacia la izquierda serán x—1 e y—1, y para el movimiento hacia abajo necesitaremos y+1 con x—1 para el movimiento hacia la izquierda, o con x+1 para moverse hacia la derecha. La figura 8.1 muestra los valores necesarios para xn e yn en las ocho direcciones, suponiendo que la pelota está en la posición x,y. En estos movimientos horizontales, verticales o diagonales, los valores de x y/o y se alteran uno tras otro.

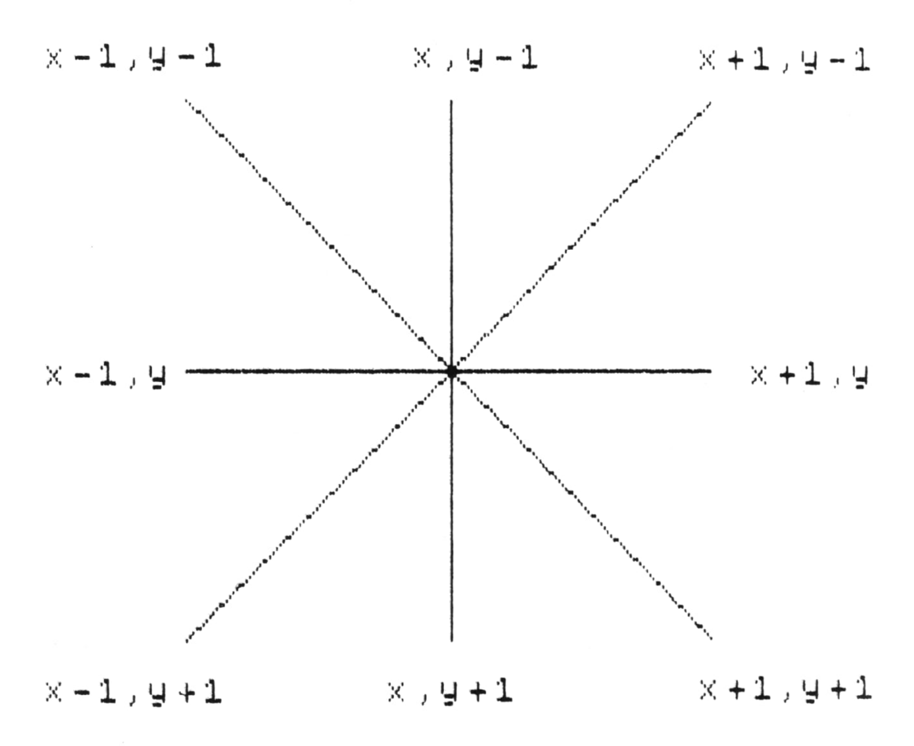


Fig. 8.1. Cambio de las coordenadas X,Y en los movimientos en diversas direcciones.

En los movimientos a mayor velocidad podemos cambiar los valores de x e y en más de una unidad por paso. Para disponer de

esta posibilidad, es conveniente utilizar dos variables más, dx y dy para representar la diferencia en las posiciones x e y por cada caso. Además, existe una ventaja adicional en el uso de los términos dx y dy. Es muy sencillo invertir la dirección del movimiento, simplemente fijando dx = —dx o dy = —dy.

Rebote en las paredes

Si intenta mover la pelota utilizando la secuencia del programa que acabamos de discutir, aparecerán problemas inmediatamente. Suponga que comenzamos con la pelota en el centro de la pantalla, y luego la vamos moviendo paso a paso hacia la derecha. Todo irá perfectamente hasta que la pelota pase el borde de la pantalla y el valor de x se haga 32. En este punto, el programa se parará y aparecerá un mensaje de error. La razón es que el valor de x se ha salido de su límite permitido, que es el valor 31. Una situación similar ocurrirá si nos vamos moviendo en la dirección opuesta y x se hace negativo. Si la pelota se mueve verticalmente, los errores aparecerán si la y se hace negativa o si es mayor que 21. Para evitar este estado de cosas, podemos establecer que cuando la pelota alcance uno de los bordes de la pantalla, se detecte su movimiento y se invierta la dirección de la moción. El efecto de esta acción es el siguiente: cuando la pelota alcance uno de los bordes de la pantalla, se reflejará en ella y saldrá despedida hacia el centro, como si rebotara desde una pared.

El rebote es una acción muy fácil de realizar. Una vez calculados los valores xn e yn para la siguiente posición de la pelota, se comparan con los valores límite para x e y. Comencemos con el valor x. En este caso, una sencilla instrucción IF comprueba el valor de xn para ver si es igual a Ø o a 31. Si la prueba resulta cierta, el signo de dx se cambia, y se hace dx = —dx. Esta acción previene que la pelota se salga de la pantalla. Se calcula un nuevo valor de xn añadiendo la nueva versión de dx a xn, para establecer el siguiente movimiento en el que la pelota se moverá en la dirección opuesta de x, hacia el centro de la pantalla. Esta técnica básica funcionará incluso si dx es mayor que 1, con tal de que la prueba IF compruebe si xn ⩽ Ø o xn > = 31, y dx se añada a xn antes de la impresión del símbolo.

El mismo proceso básico puede aplicarse al valor de yn, y en este caso dy se cambia de signo, si yn< = Ø ó yn> = 21. De nuevo, si dy cambia de signo se añade a yn antes de imprimir el símbolo. Después de estas pruebas, se borra el símbolo de la antigua posición , y la

```
100 REM Pelota en movimiento.
110 BORDER 1
120 PAPER 4
130 CLS
140 LET x=15
150 LET y=10
160 LET dx=1
170 LET dy=1
180 INK 7
190 PRINT AT y,x;CHR$ 143;
200 REM Bucle de movimiento.
210 LET xn=x+dx
220 LET yn=y+dy
230 IF xn=0 OR xn=31 THEN LET dx=-dx
240 IF yn=0 OR yn=21 THEN LET dy=-dy
250 PRINT AT y,x;" ";
260 PRINT AT yn,xn;CHR$ 143;
270 LET x=xn: LET y=yn
280 GO TO 210
```

Fig. 8.2. Símbolo de una bola que se mueve.

pelota se imprime en la nueva posición. Finalmente x e y se ponen al día con los nuevos valores, preparados para el próximo paso. Esta acción se muestra en el programa listado en la figura 8.2. Cuando se ejecute, mostrará un bloque que comienza en el centro de la pantalla y se mueve en diagonal, rebotando de los lados de la pantalla cuando los alcance.

Rebotando en un bate

En juegos tipo SQUASH o BREAKOUT la bola rebota en un bate movible y, dependiendo de donde choque la bola, la trayectoria después de la reflexión es recta, o diagonal. En este nuevo caso, necesitamos tener en cuenta la posición del bate, utilizando dos nuevas variables, bx y by.

Ahora, además de la rutina de las pruebas de reflexión, necesitamos incluir una nueva comprobación, en este caso ver si la nueva posición de la bola (xn,yn) es igual a la del bate (bx,by). Si las dos posiciones son iguales, el cambio de dirección se arregla invirtiendo dy, si el bate se encuentra en el comienzo o en el final de la pantalla, o dx, si el bate se encuentra en uno de los lados de la pantalla. Al mismo tiempo, pondremos al día la cuenta de los tantos.

Los bates son normalmente mayores que un espacio de carácter, y un tamaño típico puede ser tres caracteres de anchura. También necesitamos comprobar la posición de la bola frente a bx+1,by y

bx+2,by. En este caso suponemos que el símbolo izquierdo del bate está en la posición bx. Como de costumbre, la trayectoria de la bola después del choque viene determinada por el lugar en el que pega al bate, así pues, si toca al bate en el centro (es decir, en xn, yn = bx+1), dx se pone en ∅ y dy en —1, y la bola atraviesa la pantalla de forma rectilínea hacia arriba. Esto supone que el bate se encuentra en la parte inferior de la pantalla. Si la bola pega en uno de los otros segmentos del bate, las trayectorias son diagonales, ya que dx está en +1 o en —1 y dy en —1. En muchos juegos de este tipo, como, por ejemplo, el BREAKOUT, la pared de fondo, que contiene el bate, se comprueba para ver cuándo coincide con la pelota, y, si es así, la bola se pierde y el juego termina o se utiliza una nueva bola. En este caso la comprobación se realiza sencillamente para ver la coincidencia entre yn y 21, y si tal coincidencia existe, se hace otra comprobación con bx, para ver si la bola pegó al bate.

Movimiento del bate

La mejor manera de determinar la posición del bate en el Spectrum, es usar dos teclas adyacentes del teclado, la una da el movimiento a la derecha, y la otra a la izquierda. De hecho, pueden utilizarse las teclas de flecha, pero, por el momento, vamos a utilizar la tecla Z para movernos a la izquierda, a la tecla X para movernos a la derecha. La técnica de utilización de estos programas es similar a la utilizada en los programas de bosquejo de los Capítulos Dos y Seis.

```
100 REM Juego sencillo de squash.
110 BORDER 1
120 LET hs=0
130 LET bx=15: LET by=20
140 LET bx1=bx: LET by1=by
145 REM Establece un nuevo juego.
150 CLS
160 LET nb=0
170 LET x=15: LET y=9
180 LET sc=0
190 LET b$=CHR$ 143+CHR$ 143+CHR$ 143
200 PRINT AT 21,0;"Pulse SPACE para servicio
210 LET a$=INKEY$: IF a$<>" " THEN GO TO 210
215 REM Nuevo servicio.
220 LET x=15: LET y=9
230 PRINT AT y,x;CHR$ 143;
240 LET dx=INT (RND*3)-1: LET dy=-1
245 REM Bucle del juego principal.
```

```
250 PRINT AT 21,0;"Puntos= ";sc;
260 PRINT "     Record  = ";hs;"     ";
265 REM Mueve el bate.
270 LET m$=INKEY$
280 IF m$="z" THEN LET bx1=bx-1
290 IF m$="x" THEN LET bx1=bx+1
300 IF bx1<0 THEN LET bx1=0
310 IF bx1+2>31 THEN LET bx1=29
320 PRINT AT by,bx;"   ";
330 PRINT AT by1,bx1;b$;
340 LET bx=bx1: LET by=by1
345 REM Mueve la pelota.
350 LET xn=x+dx: LET yn=y+dy
355 REM Comprueba las paredes.
360 IF xn=0 OR xn=31 THEN LET dx=-dx
370 IF yn=0 THEN LET dy=-dy
375 REM Comprueba el fondo de la pantalla.
380 IF yn=by THEN GO TO 450
390 PRINT AT y,x;" ";
400 PRINT AT yn,xn;CHR$ 143;
405 REM Actualiza la posicion de la pelota.
410 LET x=xn: LET y=yn
420 GO TO 250
440 REM comprueba los golpes del bate.
450 IF xn=bx THEN GO TO 550
460 IF xn=bx+1 THEN GO TO 550
470 IF xn=bx+2 THEN GO TO 550
480 LET nb=nb+1
485 REM Comprueba el final del partido.
490 IF nb=5 THEN GO TO 1000
500 PRINT AT y,x;" ";
510 GO TO 200
540 REM Establece la nueva direccion.
550 LET dx=-dx: LET dy=-1
560 IF xn=bx AND bx<>0 THEN LET dx=-1
570 IF xn=bx+2 AND bx<>29 THEN LET dx=1
575 REM Actualiza la puntuacion.
580 LET sc=sc+1
590 GO TO 250
990 REM Fin del juego.
1000 BEEP 1,12
1010 PRINT AT 0,0;"FIN DEL JUEGO"
1015 REM Comprueba la puntuacion mas alta.
1020 IF hs>sc THEN GO TO 1050
1030 PRINT FLASH 1;AT 2,0;"*NUEVO  R E C O R D *"
1040 LET hs=sc
1050 INPUT "Otro juego? (s/n)";g$
1060 IF g$="s" THEN GO TO 130
1070 STOP
```

Fig. 8.3. Juego sencillo tipo squach.

Hemos llegado, pues, a un punto es el que podemos imaginar un juego sencillo donde los tantos se aumenten cada vez que la bola pegue al bate, y el juego termine después de la quinta bola. Cada nueva bola comienza desde el centro de la pantalla, con la trayecto-

ria hacia arriba. El listado de este programa se muestra en la figura 8.3. Obsérvese que, en este caso, el límite superior de la pantalla se ha fijado en y = 2, para permitir una anotación de los tantos en la línea superior.

Animación utilizando la alta resolución

Una de las desventajas que tenía la utilización de texto o baja resolución en las representaciones, era que, al ser las etapas del movimiento del objeto muy largas, el movimiento resultante es a saltos. Si pasamos a la alta resolución, la situación mejora.

En las pantallas de alta resolución, los principios básicos de movimiento de un objeto son siempre los mismos. Existe una gran diferencia (en alta resolución) entre imprimir un símbolo y trazar un punto. El valor de Y en una pantalla de texto comienza en Ø en la parte superior de la pantalla, y se va incrementando según vamos bajando por la pantalla, y, por el contrario, un punto se mueve por la pantalla hacia arriba cuando se incrementa el valor de y. Si cambiamos los valores de x e y para el objeto, produciremos diferentes direcciones de moción, como puede verse en la figura 8.4. En la pantalla de alta resolución, un sencillo paso en cualquier dirección mueve el objeto una distancia menor, y el resultado es un movimiento más suave. El programa que se lista en la figura 8.5 muestra sencillamente el movimiento de un punto en la pantalla de alta resolución.

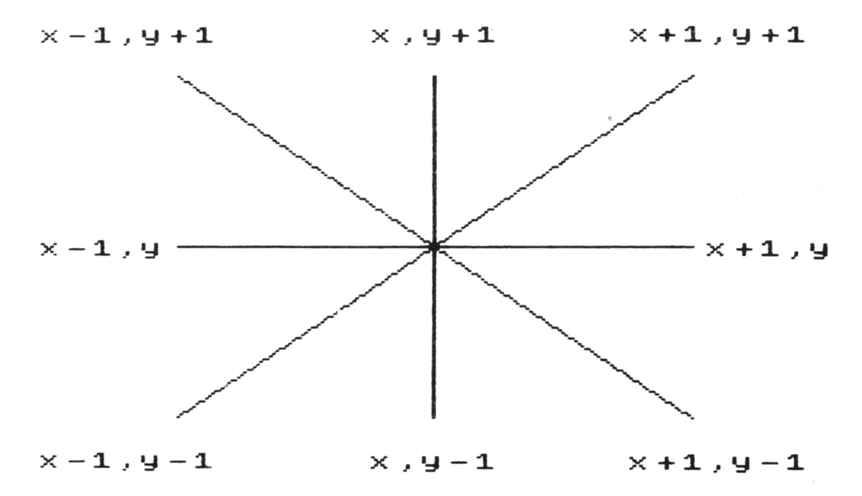


Fig. 8.4. Cambios en las coordenadas x, y necesarios para la alta resolución.

```
100 REM Movimiento de un punto en alta
resolucion.
110 BORDER 6
120 LET x=128
130 LET y=88
140 LET dx=2
150 LET dy=2
170 PLOT x,y
180 LET xn=x+dx
190 LET yn=y+dy
195 REM  Comprueba los limites de la pantalla.
200 IF xn>0 AND xn<255 THEN GO TO 220
210 LET dx=-dx: LET xn=xn+dx
220 IF yn>0 AND yn<175 THEN GO TO 240
230 LET dy=-dy: LET yn=yn+dy
235 REM  Borra el punto anterior.
240 PLOT OVER 1;x,y
245 REM Dibuja un nuevo punto.
250 PLOT OVER 1;xn,yn
260 LET x=xn
270 LET y=yn
280 GO TO 180
```

Fig. 8.5. Movimiento de un punto por la pantalla de alta resolución.

Movimientos de objetos más complejos

Hasta el momento hemos tratado de mover un punto por la pantalla, pero normalmente desearemos mover algo más complejo, como, por ejemplo, un platillo volador, Esto último puede verse en el programa listado en la figura 8.6.

```
100 REM Platillo volador en alta resolucion.
110 PLOT 0,0
120 DRAW 255,0
130 DRAW 0,175
140 DRAW -255,0
150 DRAW 0,-175
160 LET x=128: LET y=88
170 LET x1=x: LET y1=y
180 LET dx=2: LET dy=dx
190 OVER 1
200 GO SUB 500
210 FOR n=1 TO 500
220 LET x1=x+dx: LET y1=y+dy
230 IF x1+dx<6 OR x1+dx>248 THEN LET dx=-dx
240 IF y1+dy<6 OR y1+dy>168 THEN LET dy=-dy
260 GO SUB 500
270 LET x=x1: LET y=y1
280 GO SUB 500: NEXT n
290 STOP
500 PLOT x,y-2: DRAW 3,0: DRAW 2,2
510 DRAW -2,2: DRAW -1,0: DRAW -2,2
520 DRAW -2,-2: DRAW -1,0: DRAW -2,-2
530 DRAW 2,-2: DRAW 3,0: RETURN
```

Fig. 8.6. Programa del platillo volador.

El propio platillo volador se produce mediante un comando PLOT y una serie de comandos DRAW en una subrutina. El programa principal calcula las posiciones x, y para el platillo, y luego llama a la subrutina que es la que dibuja el platillo. El comando OVER1 se establece antes de comenzar el dibujo. Para borrar el platillo, sencillamente basta con volver a dibujarlo en la misma posición. En este programa el platillo se dibuja siempre en la posición x,y. Una vez calculada la posición x1,y1, el platillo se vuelve a dibujar para borrar la imagen en x,y. Más tarde, se actualizan x, y en x1,y1, y se vuelve a dibujar el platillo.

Como en el caso de la pelota y el punto, el movimiento se realiza a pasos, y éstos se establecen mediante los términos dx y dy. Una vez calculada una nueva posición, se compara con los límites de la pantalla y, si es necesario, se alteran los valores dx y dy, para que el platillo rebote desde de las fronteras de la pantalla. Obsérvese que, en este caso, los límites de la pantalla están a cierta distancia del borde de ésta para permitir que quepa la anchura y la altura del platillo. (Recuerde que la posición del platillo se mide hasta la mitad de la figura).

Un problema que observará es que el proceso de dibujos en BASIC es relativamente lento, y esto hace que este tipo de animación sea limitado, ya que su lentitud es inevitable, a no ser que usted escriba el programa en código de máquina. Por esta razón, la mayoría de los juegos de acción del Spectrum están escritos en código máquina, o al menos lo están las rutinas del movimiento.

Animación utilizando símbolos gráficos especiales.

Hay otro sistema de animación que utiliza técnicas de impresión PRINT, normales, pero con el que el movimiento es más suave. Esto implica el uso de símbolos gráficos especiales hechos a medida.

Supongamos que queremos mover un objeto en forma de diamante por la pantalla. Podemos empezar por crear el símbolo de la forma adecuada, tal y como se describe en el Capítulo Cinco. Supongamos ahora que queremos mover la figura por la pantalla en pasos de dos posiciones. En la segunda posición, parte del objeto se habrá movido al siguiente espacio de carácter. Podemos controlar todo esto muy fácilmente, creando dos nuevos símbolos que, al imprimirse uno tras otro, muestren el diamante en su nueva posición. En el siguiente paso, se crea otro par de símbolos, y el diamante se dibuja en la mitad entre las dos posiciones de los símbolos. La

cuarta posición tiene el diamante casi totalmente en el siguiente espacio de carácter. En el quinto paso, el símbolo estará en el espacio de carácter siguiente, y podemos recomenzar de nuevo el proceso, esta vez una posición más avanzada en la pantalla.

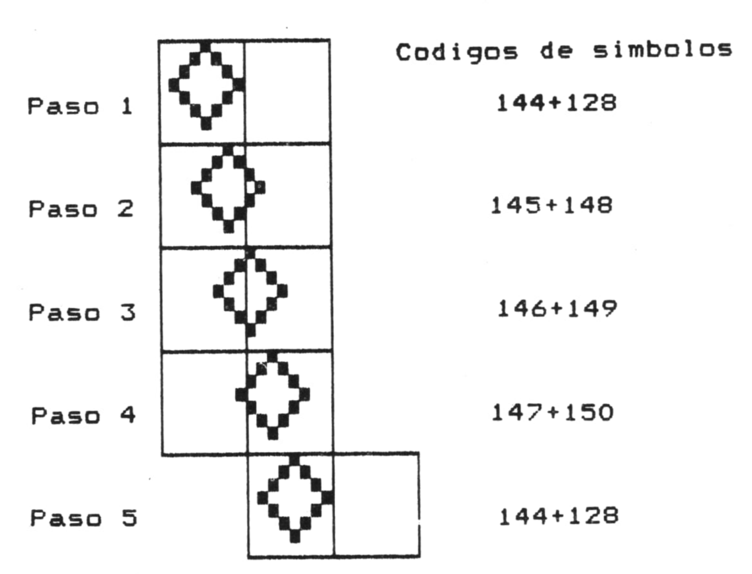


Fig. 8.7. Secuencia de pasos para mover una figura de diamante.

Esta secuencia se muestra en la figura 8.7., y el programa que mueve el diamante por la pantalla se muestra en la figura 8.8.

Esta técnica puede aplicarse rápidamente a objetos más complejos. Suponga que tenemos un platillo volador que utiliza dos espacios adyacentes en la pantalla. En este caso necesitaremos tres espacios adyacentes, para realizar la técnica. En este caso también, cuando el objeto se haya movido fuera de la posición del primer símbolo del grupo, el patrón completo se mueve un espacio de símbolo, y la acción de la animación se repite.

```
100 REM Movimiento de una figura de diamante.
110 REM Preparacion de los simbolos.
120 GO SUB 500
125 REM Bucle de movimiento.
130 FOR m=1 TO 20
140 FOR c=1 TO 20
150 PRINT AT 10,c;CHR$ 144;CHR$ 128;
160 PRINT AT 10,c;CHR$ 145;CHR$ 148;
170 PRINT AT 10,c;CHR$ 146;CHR$ 149;
180 PRINT AT 10,c;CHR$ 147;CHR$ 150;
190 PRINT AT 10,c;CHR$ 128;CHR$ 144;
200 NEXT c
210 NEXT m
```

```
220 STOP
490 REM Prepara los caracteres.
500 POKE USR "a",BIN 00001000
510 POKE USR "a"+1,BIN 00010100
520 POKE USR "a"+2,BIN 00100010
530 POKE USR "a"+3,BIN 01000001
540 POKE USR "a"+4,BIN 00100010
550 POKE USR "a"+5,BIN 00010100
560 POKE USR "a"+6,BIN 00001000
570 POKE USR "a"+7,0
580 POKE USR "b",BIN 00000010
590 POKE USR "b"+1,BIN 00000101
600 POKE USR "b"+2,BIN 00001000
610 POKE USR "b"+3,BIN 00010000
620 POKE USR "b"+4,BIN 00001000
630 POKE USR "b"+5,BIN 00000101
640 POKE USR "b"+6,BIN 00000010
650 POKE USR "b"+7,0
660 POKE USR "c",BIN 00000000
670 POKE USR "c"+1,BIN 00000001
680 POKE USR "c"+2,BIN 00000010
690 POKE USR "c"+3,BIN 00000100
700 POKE USR "c"+4,BIN 00000010
710 POKE USR "c"+5,BIN 00000001
720 POKE USR "c"+6,BIN 00000000
730 POKE USR "c"+7,0
740 POKE USR "d",0
750 POKE USR "d"+1,0
760 POKE USR "d"+2,0
770 POKE USR "d"+3,1
780 POKE USR "d"+4,0
790 POKE USR "d"+5,0
800 POKE USR "d"+6,1
805 POKE USR "d"+7,0
810 POKE USR "e",0
820 POKE USR "e"+1,0
830 POKE USR "e"+2,BIN 10000000
840 POKE USR "e"+3,BIN 01000000
850 POKE USR "e"+4,BIN 10000000
860 POKE USR "e"+5,0
870 POKE USR "e"+6,0
880 POKE USR "e"+7,0
890 POKE USR "f",BIN 10000000
900 POKE USR "f"+1,BIN 01000000
910 POKE USR "f"+2,BIN 00100000
920 POKE USR "f"+3,BIN 00010000
930 POKE USR "f"+4,BIN 0010000
940 POKE USR "f"+5,BIN 01000000
950 POKE USR "f"+6,BIN 10000000
960 POKE USR "f"+7,0
970 POKE USR "g",BIN 00100000
980 POKE USR "g"+1,BIN 01010000
990 POKE USR "g"+2,BIN 10001000
1000 POKE USR "g"+3,BIN 00000100
1010 POKE USR "g"+4,BIN 10001000
1020 POKE USR "g"+5,BIN 01010000
1030 POKE USR "g"+6,BIN 00100000
1040 POKE USR "g"+7,0
1050 RETURN
```

Fig. 8.8. Programa que dibuja un diamante moviéndose.

Colisiones con otros objetos

En muchos programas tipo juegos, como, por ejemplo, los invasores espaciales, el objetivo es lanzar misiles, o tirar bombas sobre otros objetos y destruirlos. Esto significa que debemos detectar cuándo los misiles alcanzan la misma posición que el objeto que hace el blanco. Una técnica que puede usarse para ello es mantener una tabla (x,y) de las posiciones de los objetos en la pantalla. Antes de mover el misil a su nueva posición, se comprueba, por comparación, su posición con la de los otros objetos, uno tras otro. Si alguna concuerda, el programa irá a una subrutina o procedimiento que produce la necesaria explosión, y se borran ambos, el misil y el blanco. Si la pantalla se llena de objetos, la situación puede complicarse mucho más.

Podemos utilizar una técnica más sencilla para detectar el momento en el que un misil choca con un objeto. Lo podemos hacer utilizando el comanco POINT para comprobar si la siguiente posición a la que debe moverse el misil tiene ya el color INK (de texto). Si no es así, el misil se mueve de forma normal. Si el comando POINT detecta un punto en el color INK, el misil está a punto de chocar contra un objeto, y el programa llamará a una rutina de impactos. Entonces se borrarán el misil y el blanco, y se sustituye el objeto por un efecto de explosión, adecuadamente acompañada por sonido, por supuesto. Finalmente la imagen de la explosión se borra y comienza un nuevo juego.

Animación que implica cambios de forma

Hasta ahora, en nuestros experimentos con la animación, el objeto que se movía tenía siempre la misma forma en su movimiento por toda la pantalla. Esto, por supuesto, está muy bien para objetos como pelotas, platillos voladores, y cosas semejantes. Si visualizamos extranjeros, u hombres, la situación es diferente.

Si dibujamos la figura de un monigote y la movemos por la pantalla del mismo modo que movemos el platillo, parecerá que el hombre se desliza por la pantalla, ya que sus piernas y sus brazos no cambian de posición durante el movimiento. Incluso los extranjeros del juego típico de los invasores mueven sus piernas cuando se mueven por la pantalla.

La técnica para producir cambios en la forma de un objeto exige la utilización de dos o más versiones del objeto que se va a animar.

Comencemos por tomar un extranjero invasor relativamente sencillo. Queremos que las piernas se muevan, y el movimiento más sencillo puede ser tener las piernas hacia adentro en un paso, y tenerlas hacia afuera en el siguiente. El primer paso, evidentemente, es crear símbolos especiales para dos dibujos separados de nuestro extranjero, uno de ellos con las piernas juntas, y el otro con ellas separadas.

Para producir ahora la animación que necesitamos, dibujaremos la primera de las dos formas.

Lo primero es calcular la posición siguiente del extranjero, y luego borrar la figura anterior y dibujar la segunda forma, en la nueva posición. Para el paso siguiente, dibujamos la primera forma de nuevo, y así sucesivamente, alternando las formas según el extranjero se vaya moviendo por la pantalla. Este programa se muestra listado en la figura 8.9.

Para el caso de un hombre que anda, el nivel de complejidad es superior. En este caso, se crean cuatro formas separadas, y organizadas como una cadena. De nuevo, cuando el hombre anda por la pantalla, los cuatro dibujos se van trazando y borrando en secuencia para producir el efecto de un hombre que anda. Pueden obtenerse mejores resultados utilizando más dibujos intermedios, que vayan formando cada etapa que realiza el hombre. En este caso, hemos cambiado el uso de muchas imágenes para obtener un movimiento más suave, por la velocidad y la utilización de menos memoria. Cuantas más etapas haya, más le costará al hombre realizarlas. Puede llegarse a un compromiso en el que la acción sea razonablemente real, pero no demasiado compleja o demasiado lenta. Recuerde que un objeto que se mueve deprisa no necesita que su acción esté dibujada con tanto detalle, porque la velocidad cubre muchas de las imperfecciones del cambio de forma que se utilizó.

La animación de objetos que cambian de forma, especialmente si son objetos de la vida diaria, familiar, implica un estudio del movimiento de estos objetos en la vida real. Más tarde, utilizaremos una versión simplificada para animar el objeto dibujado con el ordenador. De hecho, el proceso de animación es una forma de arte en sí misma, y la experimentación con diferentes ideas puede resultar muy divertida. En nuestro caso, hemos resaltado los principios más interesantes y algunas de las técnicas usadas en la animación de objetos en la pantalla gráfica del ordenador.

```
100 REM Movimiento de un extraterrestre
110 REM con cambio de figura.
120 REM Prepara los simbolos
130 GO SUB 500
135 REM Secuencia de animacion.
140 FOR p=20 TO 0 STEP -2
150 LET r=10
160 FOR c=0 TO 28 STEP 2
170 PRINT AT r,c;CHR$ 144;CHR$ 145;
180 PAUSE p
190 PRINT AT r,c;"  ";
200 PRINT AT r,c+1;CHR$ 146;CHR$ 147;
210 PAUSE p
220 PRINT AT r,c+1;"  ";
230 NEXT c
240 NEXT p
250 STOP
490 REM Prepara los caracteres.
500 POKE USR "a",BIN 00111110
510 POKE USR "a"+1,BIN 01000001
520 POKE USR "a"+2,BIN 01011100
530 POKE USR "a"+3,BIN 01000000
540 POKE USR "a"+4,BIN 00111111
550 POKE USR "a"+5,BIN 00100100
560 POKE USR "a"+6,BIN 01001000
570 POKE USR "a"+7,BIN 10010000
580 POKE USR "b",BIN 01111100
590 POKE USR "b"+1,BIN 10000100
600 POKE USR "b"+2,BIN 00111010
610 POKE USR "b"+3,BIN 00000010
620 POKE USR "d"+4,BIN 11111100
630 POKE USR "b"+5,BIN 00100100
640 POKE USR "b"+6,BIN 01001000
650 POKE USR "b"+7,BIN 10010000
660 POKE USR "c",BIN 00111110
670 POKE USR "c"+1,BIN 01000001
680 POKE USR "c"+2,BIN 01011100
690 POKE USR "c"+3,BIN 01000000
700 POKE USR "c"+4,BIN 00111111
710 POKE USR "c"+5,BIN 00100100
720 POKE USR "c"+6,BIN 00011000
730 POKE USR "c"+7,BIN 00100100
740 POKE USR "d",BIN 01111100
750 POKE USR "d"+1,BIN 10000100
760 POKE USR "d"+2,BIN 00111010
770 POKE USR "d"+3,BIN 00000010
780 POKE USR "d"+4,BIN 11111100
790 POKE USR "d"+5,BIN 00100100
800 POKE USR "d"+6,BIN 00011000
810 POKE USR "d"+7,BIN 00100100
820 RETURN
```

Fig. 8.9. Programa que dibuja una figura de extraterrestre invasor.

Capítulo 9
Profundidad y perspectiva

Los gráficos y diagramas que hemos dibujado hasta ahora tenían sólo dos variables, x e y, y se trazaban horizontalmente y verticalmente en la pantalla. Sin embargo, en la vida real, en muchas ocasiones son necesarias tres variables. La tercera variable suele llamarse z. Un ejemplo de este problema puede ser el caso en el que deseemos saber la altura de unos puntos en una pequeña franja de tierra. En este caso, nuestras coordenadas x e y representarían, por ejemplo, la anchura y la profundidad, y localizarían cualquier punto en la superficie de la tierra. El tercer término, z, será la altura de la superficie de la tierra en ese punto. En este caso, el valor de z dependerá de ambas variables, x e y, ya que cualquier cambio en una cualquiera de ellas, x ó y, nos llevaría a otro punto de la superficie de la tierra, con un valor diferente de z.

El dibujo de gráficos con tres ejes requiere técnicas ligeramente diferentes, ya que necesitamos encontrar el modo de alojar el eje Z. Si X e Y se trazan en la pantalla como siempre, teóricamente, las coordenadas Z se saldrían de la pantalla. Evidentemente esto no es posible en absoluto, y tendremos que buscar otros esquemas.

Una posibilidad es trazar Z con X en la pantalla, y dar una serie de gráficos para los diferentes valores de Y. Si estos gráficos se colocan uno sobre otro, el resultado no es muy satisfactorio.

Suponga que estamos construyendo un modelo de dibujo con tres ejes, en cartulina. El primer paso sería trazar una serie de gráficos con los ejes Z y X. Una vez trazados, el siguiente paso es colocar los gráficos uno detrás de otro. ¿Acaso podemos hacerlo en nuestra pantalla?

Una solución para visualizar estas series de gráficos puede ser dibujarlas de forma que cada valor de Y se desplace a la izquierda y hacia arriba en la pantalla. De este modo separamos los gráficos individuales para los diferentes valores de Y. En efecto, estamos dibujando el eje Y a lo largo de una línea que va hacia arriba y a la

izquierda del origen X,Y,Z, (en él las tres coordenadas son cero). Esto mejora los gráficos, pero sigue sin resolver el problema.

La solución usual es presentar los tres ejes, con el X y el Y formando 30 grados con la horizontal y el eje Z vertical, como puede verse en la figura 9.1. Ahora, cuando la X se incrementa, el punto se mueve hacia arriba y a la derecha, y, si se incrementa la Y, el punto se mueve hacia arriba y a la izquierda. Finalmente, Z desplaza el punto sólo verticalmente por la pantalla.

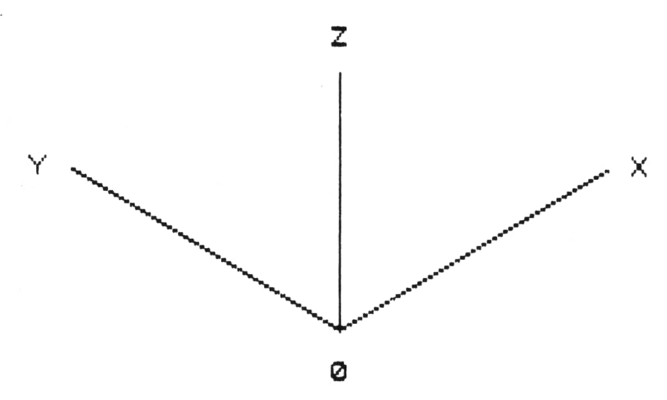


Fig. 9.1. Presentación con los tres ejes, X, Y y Z.

Diagramas de barras con tres ejes

Un tipo de diseño que impresiona es el diagrama de barras en tres dimensiones. El primer peldaño para construir este tipo de diagrama es elegir un punto de origen donde los valores X,Y y Z valgan 0. Este punto determina dónde se visualizará el diagrama de barras, y también actúan como punto de referencia alrededor del cual se construirá el diseño. El siguiente paso será dibujar una rejilla mostrando las coordenadas X é Y para el plano en el que Z = ∅.

Para dibujar el eje X en el ángulo deseado, necesitamos mover la coordenada Y la mitad del movimiento de la X, y, por tanto, nuestras coordenadas para los puntos a lo largo del eje X (Y y Z ambas ∅) serán:

X1 = CX = X
Y1 = CY = X/2

Cuando X y Z son las dos ∅, la línea que representa el eje Y deberá ir hacia arriba y a la izquierda. Como el movimiento es a la izquierda

del origen, (CX, CY) significa que las coordenadas X de pantalla, para puntos a lo largo del eje Y, deben ser inferiores a CX. Para obtener el ángulo de 30 grados a la izquierda, el movimiento de X se hace igual al de la Y, pero negativo, y la mitad. Las coordenadas de pantalla se hacen, por tanto:

$$X1 = CX + X$$
$$Y1 = CY + X/2$$

Cuando las coordenadas X y Z se hacen 0, la línea de eje Y va hacia arriba y hacia la izquierda. Como el movimiento es hacia la izquierda del origen (CX,CY), esto significa que la coordenada X de pantalla para los puntos a lo largo del eje Y debe ser inferior a CX. Para obtener el ángulo de 3Ø grados a la izquierda, el movimiento de la X se hace igual al movimiento de la Y, pero negativo, y el movimiento de la Y se hace la mitad. Por tanto, las coordenadas de pantalla serán:

$$X = CX - Y$$
$$Y1 = CY + Y/2$$

Para cualquier otro punto sobre el plano Z = 0, la posición de X1, 1 se producirá combinando los dos resultados que obtuvimos antes, para dar:

$$X1 = CX + X - Y$$
$$Y1 = CY + X/2 + Y/2$$

Estamos en un punto en el que ya podemos confeccionar un trozo de programa con todo lo obtenido. Este programa se lista en la figura 9.2. y produce un dibujo del plano X,Y del gráfico para Z = Ø, tal y como se muestra en la figura 9.3.

El término Z se traza verticalmente y, por tanto, sólo afectará al valor Y de un punto del diagrama. Como vamos a dibujar una línea vertical representando a la coordenada Z, necesitamos saber las coordenadas del final de la línea. En este caso, el valor X es el mismo que X1, y para el nuevo valor de Y, se le añade el valor Z, por tanto, los valores de las coordenadas X2,Y2 serán:

$$X2 = X1 = CX + X - Y$$
$$Y2 = Y1 + Z = CY + X/2 + Y/2 + Z$$

```
100 REM Grafico de tres ejes para z=0
110 CLS
115 REM Establece el origen.
120 LET cx=116: LET cy=20
125 REM Marca los valores maximos de X e Y.
130 LET xm=96: LET ym=80
135 REM Establece la separacion de marcas
en los ejes.
140 LET xs=12: LET ys=8
145 REM Dibuja el eje de las X.
150 FOR y=0 TO ym STEP ys
160 LET x1=-y
170 LET x2=xm-y
180 LET y1=y/2
190 LET y2=(xm+y)/2
200 PLOT INT (cx+x1),INT (cy+y1)
210 DRAW INT (x2-x1),INT (y2-y1)
220 NEXT y
225 REM Dibuja las lineas del eje Y.
230 FOR x=0 TO xm STEP xs
240 LET x1=x
250 LET x2=x-ym
260 LET y1=x/2
270 LET y2=(x+ym)/2
280 PLOT INT (cx+x1),INT (cy+y1)
290 DRAW INT (x2-x1),INT (y2-y1)
300 NEXT x
305 REM Inserta las marcas de la escala
310 PLOT cx,cy
320 DRAW INT (xm+xs),INT ((xm+xs)/2)
330 PLOT cx,cy
340 DRAW -INT (ym+ys),INT ((ym+ys )/2)
350 PRINT AT 13,3;"y";
360 PRINT AT 12,28;"x";
370 PRINT AT 20,14;"0";
```

Fig. 9.2. El plano XY para Z=0.

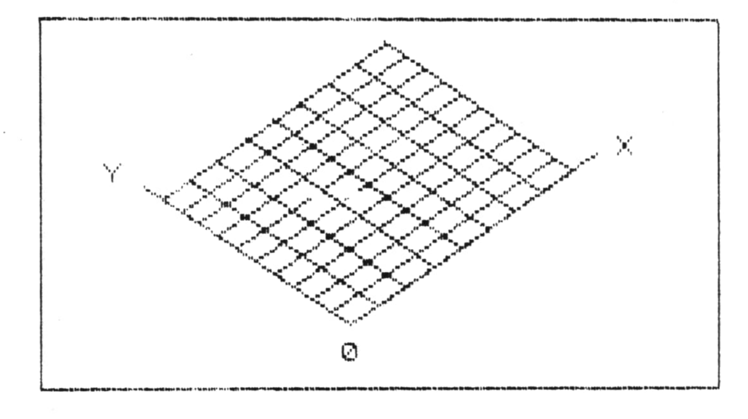


Fig. 9.3. Visualización del plano X,Y.

La coordenada Z puede obtenerse ahora dibujando líneas que comienzan en X1 Y1 y se dirigen al punto X2,Y2, utilizando el comando PLOT para llegar a X1, Y1, y con un comando DRAW para producir la línea. Como ejemplo, supongamos que queremos dibujar un gráfico con $Z = (x^2)/3 + (y^2)/2$.

Para dibujar cada punto Z, colocamos una línea vertical cuya longitud representa Z y cuyo punto de partida es el punto adecuado en el plano XY. En la figura 9.4 se puede ver un programa que dibuja este gráfico.

El resultado es un patrón de líneas verticales como un colchón de clavos, donde la altura de cada línea indica el valor de Z. Para evitar que las coordenadas Z se entremezclen, los pasos en los dos ejes X e Y y deben ser diferentes. En la figura 9.5 puede verse el resultado en la pantalla.

```
100 REM Grafico de tres ejes.
110 CLS
115 REM Estalece el origen.
120 LET cx=116: LET cy=20
125 REM Fija los valores maximos de X e Y.
130 LET xm=96: LET ym=80
135 REM Fija los saltos para las X y las Y.
140 LET xs=16: LET ys=10
150 DIM z(7,9)
155 REM Calcula y traza los valores Z.
160 FOR y=1 TO 9
170 FOR x=1 TO 7
180 LET x1=x-1: LET y1=y-1
190 LET z(x,y)=x1*x1/3+y1*y1/2
200 LET x2=INT (xs*x1-ys*y1)
210 LET y2=INT ((xs*x1+ys*y1)/2)
220 PLOT cx+x2,cy+y2
230 DRAW 0,z(x,y)
240 NEXT x
250 NEXT y
255 REM Inserta las marcas de la escala.
260 PLOT cx,cy
270 DRAW INT (xm+xs),INT ((xm+xs)/2)
280 PLOT cx,cy
290 DRAW INT -(ym+ys),INT ((ym+ys)/2)
295 REM Rotula los ejes.
300 PRINT AT 13,2;"Y";
310 PRINT AT 12,29;"X";
320 PRINT AT 20,14;"O";
330 PLOT cx,cy
340 STOP
```

Fig. 9.4. Trazado sencillo de los tres ejes.

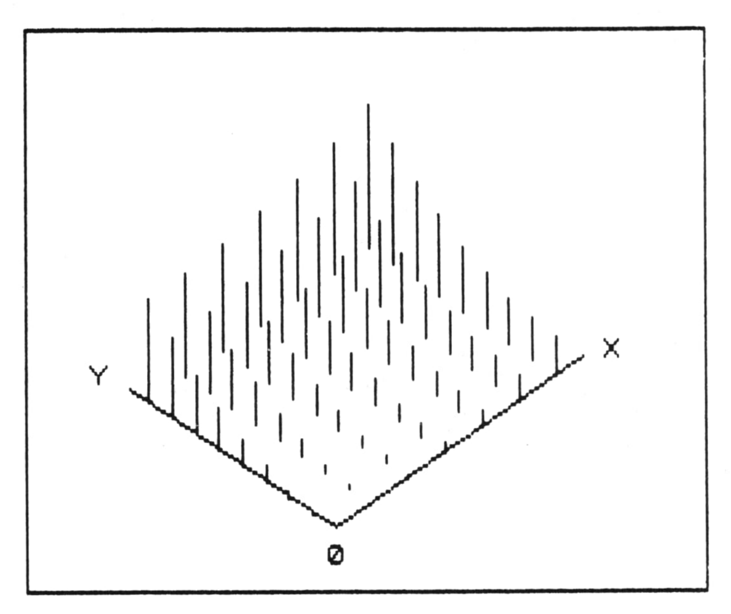


Fig. 9.5. Visualización sencilla de tres ejes.

Producción de barras anchas

En lo que respecta a los diagramas con tres ejes, hasta el momento hemos producido únicamente una línea sencilla por cada coordenada Z. El dibujo puede quedar más atractivo cambiando la sencilla línea vertical por una barra alineada, por ejemplo, con el eje X. Los cambios en el programa son muy sencillos, ya que habría que dibujar otra ordenada Z de la misma altura, pero en una posición algo diferente, y luego tendríamos que unir la parte superior y la inferior de esas dos ordenadas con dos líneas cortas. El interior de la "caja" que hemos producido, se llenaría con un color de texto (INK). Tomando el mismo conjunto de valores Z que antes, el programa revisado toma la forma del que se muestra en la figura 9.6.

Las barras se espacian la misma distancia entre ellas que su propia anchura. La parte superior y la inferior se dibujan con una línea paralela al eje X. Observe que ambas coordenadas, X e Y, se decrementan desde el máximo a cero, de forma que las barras de la parte posterior del gráfico se dibujan primero. Después de llenar de color cada barra, ésta se bordea con el color del fondo, utilizando para ello el comando INVERSE 1, antes de rebordear la barra. Después de rebordear la barra con el color de fondo (PAPER), se utiliza el color INVERSE Ø para restaurar el modo de dibujo y trazado normal. Con ello, las barras de delante resaltan y se ven

```
100 REM Diagrama en tres dimensiones
con barras anchas.
110 REM z=(4+2*SIN (x/20))*(y/10+1)
120 LET cx=128: LET cy=16
130 LET dx=10: LET dy=5
135 REM Dibuja la rejilla x,y.
140 GO SUB 400
145 REM Traza el diagrama.
150 FOR x=100 TO 0 STEP -20
160 FOR y=90 TO 0 STEP -15
170 LET z=INT ((4+2*SIN (x/20))*(y/10+1))
180 GO SUB 500
190 NEXT y
200 NEXT x
210 STOP
390 REM Subrutina de la rejilla x,y.
400 FOR x=0 TO 100 STEP 20
410 PLOT cx+x,cy+x/2
420 DRAW -100,50
430 NEXT x
440 FOR y=0 TO 90 STEP 15
450 PLOT cx-y,cy+y/2
460 DRAW 110,55
470 NEXT y
480 RETURN
500 REM Subrutina de dibujo de las barras
510 FOR n=0 TO z-1
520 PLOT cx+x-y,cy+x/2+y/2+n
530 DRAW dx,dy
540 NEXT n
545 REM Borra el perimetro de las barras.
550 INVERSE 1
560 PLOT cx+x-y,cy+x/2+y/2
570 DRAW dx,dy
580 DRAW 0,z
590 DRAW -dx,-dy
600 DRAW 0,-z
610 INVERSE 0
620 RETURN
```

Fig. 9.6. Diagrama de barras anchas con tres ejes.

cuando solapan otra barra, tal y como puede apreciarse en la figura 9.7.

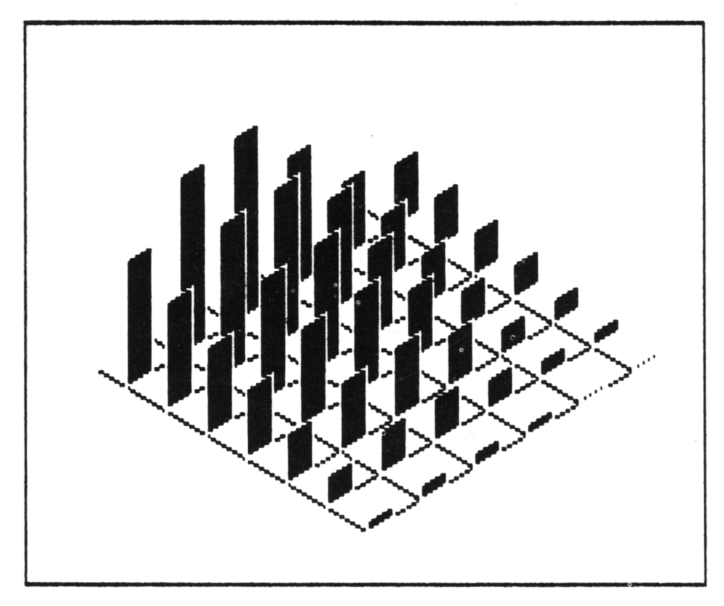


Fig. 9.7. Visualización en pantalla del diagrama de barras anchas.

Producción de barras con volumen

Un desarrollo posterior de este tipo de diagramas es el dibujo de las barras, de modo que parezcan con volumen. En efecto, dibujaremos una barra alineada con el eje X y otra alineada con el eje Y para cada coordenada Z. Finalmente, dibujaremos un rombo para la parte superior y habremos completado la barra. Uno de los lados de la barra puede colorearse si se desea.

```
100 REM Diagrama de barras con volumen utili-
105 REM zando los tres ejes.
110 REM z=(4+2*COS (x/20))*(y/10+1)
120 LET cx=128: LET cy=16
130 LET dx1=10: LET dy1=5
140 LET dx2=8: LET dy2=4
150 REM Dibuja la rejilla x,y.
160 GO SUB 400
165 REM Dibuja el diagrama.
170 FOR x=100 TO 0 STEP -20
180 FOR y=90 TO 0 STEP -15
190 LET z=INT ((4+2*COS (x/20))*(y/10+1))
195 REM Traza la barra.
200 GO SUB 500
210 NEXT y
220 NEXT x
230 STOP
```

```
390 REM Subrutina de la rejilla x,y.
400 FOR x=0 TO 100 STEP 20
410 PLOT cx+x,cy+x/2
420 DRAW -100,50
430 NEXT x
440 FOR y=0 TO 90 STEP 15
450 PLOT cx-y,cy+y/2
460 DRAW 110,55
470 NEXT y
480 RETURN
500 REM Subrutina de dibujo de las barras.
505 REM Dibuja la cara de la barra.
510 FOR n=0 TO z-1
520 PLOT cx+x-y,cy+x/2+y/2+n
530 DRAW dx1,dy1
540 NEXT n
545 REM Borra el costado de las barras.
550 INVERSE 1
560 FOR n=0 TO z-1
570 PLOT cx+x-y,cy+x/2+y/2+n
580 DRAW -dx2,dy2
590 NEXT n
600 INVERSE 0
605 REM Traza el lado de la barra.
610 PLOT cx+x-y,cy+x/2+y/2
620 DRAW -dx2,dy2
630 DRAW 0,z
640 DRAW dx2,-dy2
650 DRAW 0,-z
660 REM Borra la parte superior de la barra.
670 INVERSE 1
680 FOR n=0 TO dx2-1
690 PLOT cx+x-y-n,cy+z+(x+y+n)/2
700 DRAW dx1,dy1
710 NEXT n
720 INVERSE 0
725 REM Dibuja la parte superior de la barra.
730 PLOT cx+x-y,cy+x/2+y/2+z
740 DRAW dx1,dy1
750 DRAW -dx2,dy2
760 DRAW -dx1,-dy1
770 DRAW dx2,-dy2
780 RETURN
```

Fig. 9.8. Diagrama de barras con volumen, utilizando tres ejes.

El programa listado de la figura 9.8. produce un ejemplo de este tipo de visualización, y el resultado en la pantalla puede verse en la figura 9.9.

En el programa, la cara frontal de la barra se dibuja primero y se llena con el color INK. La siguiente etapa es dibujar el lado de la barra que sigue el eje Y, utilizando el comando INVERSE 1 que borra cualquier barra que se encuentre detrás de la que se está trazando. INVERSE ∅ se usa después para restaurar el dibujo normal, y más tarde se traza el reborde del lado de la barra. Finalmente

se borra, utilizando el comando INVERSE el rombo de la parte superior de la barra. A continuación se realiza su reborde.

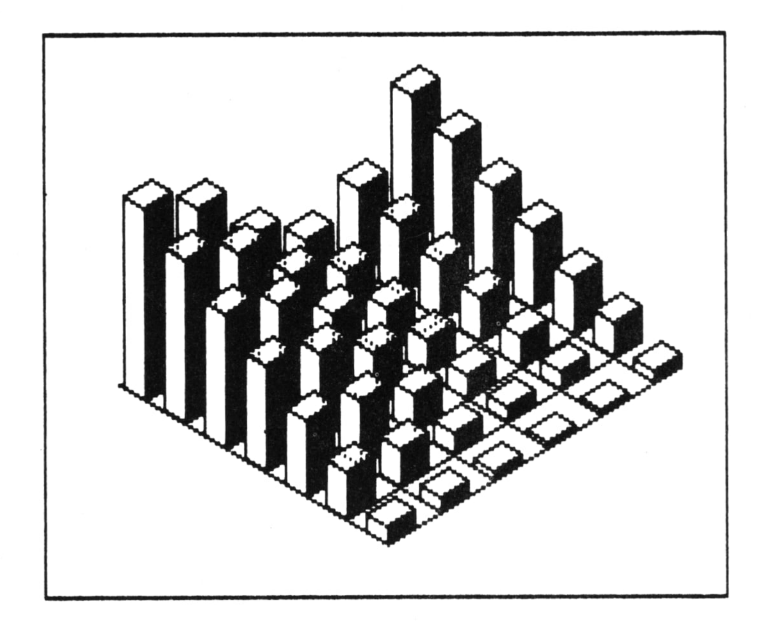


Fig. 9.9. Visualización de un diagrama de barras con volumen, utilizando los tres ejes coordenados.

Diagrama de barras X-Y con volumen

También puede dibujarse un diagrama en dos dimensiones, X,Y con barras con pseudo volumen. En esta caso, el eje X está inclinado 30 grados, y el fondo de las barras se dibuja primero. Las barras se dibujan en la parte superior, utilizando medio paso de X para el grosor y anchura de las barras. La técnica para el dibujo de las barras es exactamente la misma que la usada para el diagrama de barras con volumen con tres ejes.

En la figura 9.10. puede verse un programa que produce este tipo de visualización. Su resultado en pantalla puede verse en la figura 9.11.

```
100 REM Diagrama de barras con volumen
110 REM para y=x*x/100.
120 LET cx=64: LET cy=16
130 LET ax=6: LET ay=3
140 LET bx=8: LET by=4
150 LET lx=100: LET ly=100
160 REM Traza el plano posterior.
170 GO SUB 400
180 REM Dibuja el grafico.
190 FOR x=100 TO 0 STEP -15
200 LET y=x*x/100
205 REM Traza las barras.
210 GO SUB 500
220 NEXT x
230 STOP
390 REM Subrutina de trazado del plano posterior.
400 PLOT cx-bx,cy+by
410 DRAW lx,lx/2
420 DRAW 0,ly
430 DRAW -lx,-lx/2
440 DRAW 0,-ly
450 RETURN
500 REM Subrutina de dibujo de las barras.
505 REM Dibuja el frente de la barra.
510 FOR n=0 TO y-1
520 PLOT cx+x,cy+x/2+n
530 DRAW ax,ay
540 NEXT n
545 REM Borra el costado de las barras.
550 INVERSE 1
560 FOR n=0 TO y-1
570 PLOT cx+x,cy+x/2+n
580 DRAW -bx,by
590 NEXT n
600 INVERSE 0
605 REM Traza el costado de la barra.
610 PLOT cx+x,cy+x/2
620 DRAW -bx,by
630 DRAW 0,y
640 DRAW bx,-by
650 DRAW 0,-y
660 REM Borra la parte superior de la barra.
670 INVERSE 1
680 FOR n=0 TO bx-1
690 PLOT cx+x-n,cy+y+x/2+n/2
700 DRAW ax,ay
710 NEXT n
720 INVERSE 0
725 REM Dibuja la parte superior de la barra.
730 PLOT cx+x,cy+x/2+y
740 DRAW ax,ay
750 DRAW -bx,by
760 DRAW -ax,-ay
770 DRAW bx,-by
780 RETURN
```

Fig. 9.10. Diagrama de barras con volumen, utilizando sólo dos ejes coordenados.

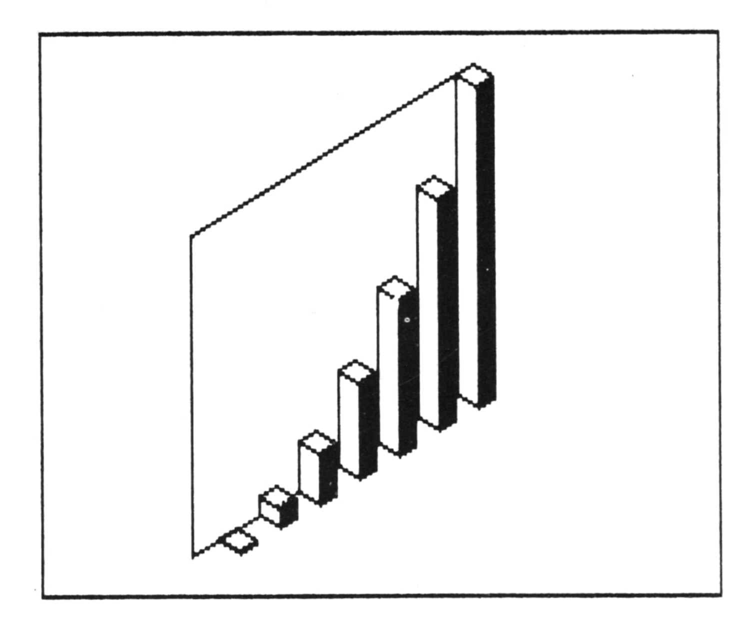


Fig. 9.11. Visualización de un diagrama de barras con volumen que utiliza dos ejes.

Mapas de superficie

En lugar de dibujar líneas verticales para las coordenadas Z, podemos trazar un dibujo que muestre los contornos de la superficie formada con los finales de las coordenadas Z. Tomaremos todas las coordenadas Z con Y = 0, y dibujaremos las líneas que las unen a todas. Luego repetiremos el proceso para cada valor de Y, para obtener una serie de líneas diagonales apuntando a la izquierda. A continuación, tomamos todas las coordenadas Z para X = 0, y volvemos a unirlas con líneas. Esto lo repetiremos para todos los valores de X, y, finalmente, tendremos un patrón que nos dará la forma en que varía Z sobre el plano XY. De nuevo podemos utilizar el color para resaltar las diferentes tiras que aparecen en plano.

El resultado obtenido, utilizando bien las coordenadas verticales Z o los puntos encadenados, será mejor si hay un gran número de puntos, ya que de este modo se obtendrá un contorno de la figura mucho más suave. Sin embargo, si se usan muchos puntos, el efecto de volumen de la figura tiende a perderse en algunos puntos que se superponen con otros, y se produce cierta confusión. Los mejores resultados se obtienen, generalmente, estableciendo la secuencia de dibujo para que empiece en puntos por detrás del plano XY, es decir, para los más altos valores tanto de X como de Y. De este modo,

cualquier línea que solape, quedará sobreescrita por la línea que se encuentre delante.

Trazados circulares con tres ejes

Una variación muy interesante del dibujo con tres ejes, es el circular, que produce unos diseños muy atractivos.

En esta versión de un diagrama con tres ejes, el plano XY del diagrama se hace elíptico, de forma que el dibujo resultante en pantalla es una especie de disco visto desde cierto ángulo, con bordes elípticos creados utilizando las coordenadas Z.

La técnica de dibujo de este tipo de gráficos utiliza el método cuadrático de dibujo de círculos para la producción de los ejes X, Y. La escala X se fija en el doble o el triple que la escala Y, para producir una elipse en la pantalla. Los valores de Z simplemente se añaden a las coordenadas Y calculadas, y se trazan puntos en las posiciones superiores de las coordenadas Z.

El programa de la figura 9.12. es un ejemplo de este tipo de trazado. La función utilizada determinará la forma del contorno de este tipo de dibujo, dándole a usted ahora la oportunidad de probar cantidad de funciones. Una visualización típica es la de la figura 9.13. Observe que este tipo de diseño toma mucho tiempo en generarse, ya que tiene muchos puntos, y los cálculos para cada uno de ellos son relativamente complejos.

```
100 REM Grafico circular en tres dimensiones.
110 REM Preparacion de la funcion.
120 LET k=PI/2000
130 LET m=1/SQR (2)
140 DEF FN a(z)=10*COS (k*(x*x+y*y))
150 REM Dibuja el grafico.
160 FOR x=-100 TO 100
170 LET y1=5*INT (SQR (10000-x*x)/5)
180 FOR y=y1 TO -y1 STEP -5
190 LET z=FN a(SQR (x*x+y*y))-m*y
195 REM Eliminacion de las lineas ocultas.
200 IF y=y1 THEN GO TO 220
210 IF z<z1 THEN GO TO 240
220 PLOT 128+x,80+INT (z/2)
230 LET z1=z
240 NEXT y
250 NEXT x
```

Fig. 9.12. Trazado de una figura circular en tres dimensiones.

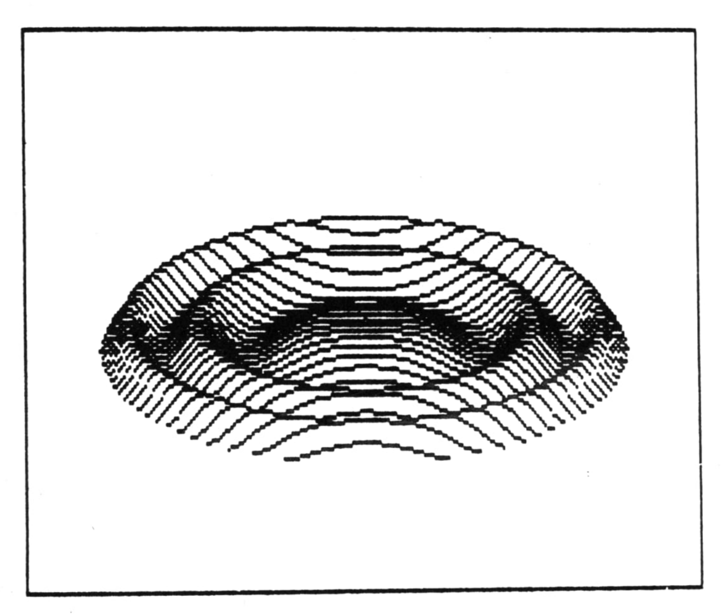


Fig. 9.13. Visualización del diagrama circular anterior.

Dibujos con perspectiva

Los diagramas y gráficos de tres ejes que hemos realizado hasta ahora dan cierta impresión de profundidad, pero a un artista le parecerían falsos. A este tipo de dibujos, se les denomina ISOMETRICOS. Esto significa básicamente que la línea vertical siempre tiene el mismo factor de escala que cualquier punto del plano XY. Este tipo de dibujos es el producido por los delineantes, ya que el objeto puede medirse con exactitud según los tres ejes.

Para dar sensación de profundidad, los artistas utilizan perspectiva en sus dibujos. Hace cientos de años que descubrieron que un objeto que se mueve lejos del observador parece menor, y viceversa. También encontraron que, al aplicar esta idea a sus dibujos y pinturas, éstas tenían un aspecto mucho más real. A esta técnica se le ha dado el nombre de dibujo en perspectiva, y hace ya muchos siglos que los matemáticos se han ocupado de encontrar la fórmula que permita calcular la forma y tamaño de los objetos para tener una visión en perspectiva correcta. Esta técnica también puede aplicarse a los gráficos del ordenador para producir visualizaciones en pantalla mucho más reales.

Para ver el efecto de la perspectiva, imagínese de pie sobre una llanura plana, con una carretera que sigue hasta el horizonte. Aun-

que los lados de la carretera son paralelos, a usted le parecerá que la carretera se va estrechando según se va acercando hacia el horizonte. Los camiones y coches que viajan por la carretera también le parecerán menores según se van alejando de su posición hacia el horizonte. De hecho, la imagen óptica que producen es menor según se van alejando de su posición hacia el horizonte. Si aplicamos a nuestros diseños en la pantalla esta regla básica, también podremos dar sensación de profundidad, a pesar de que nuestro sistema de visualización es sencillamente una pantalla plana.

Lo primero es decidir el sistema de coordenadas con el que podamos medir las posiciones de los puntos en los objetos que vemos, y los puntos correspondientes para producir la imagen en la pantalla. Supondremos que el eje X va de izquierda a derecha, como normalmente. El eje Z es vertical, como hasta ahora en nuestros gráficos de tres ejes. Ya sólo nos queda el eje Y, y lo mejor es tenerlo a lo largo de la dirección de la vista.

Si usted mira a la carretera a través del desierto, desde el nivel de la misma carretera (usted se encuentra en la superficie de ésta), su visión no le indicará nada, ya que todos los puntos de la carretera y del desierto estarán en una línea sobre el eje X. Para mirar adecuadamente la carretera, debemos colocarnos sobre ella.

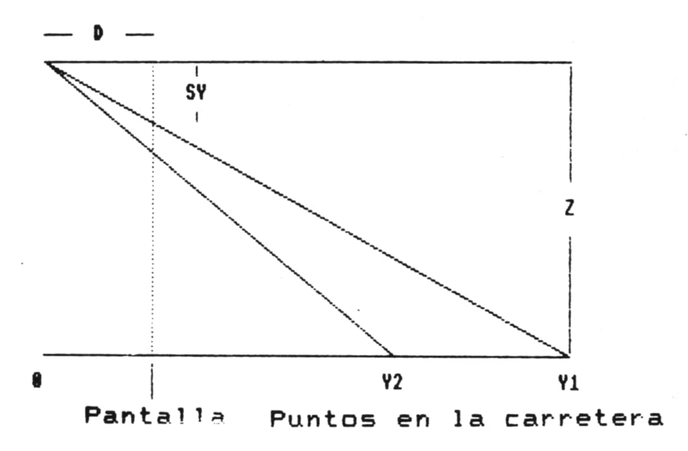


Fig. 9.14. Diagrama que muestra la relación entre la carretera y la pantalla.

La figura 9.14 muestra una vista lateral de la situación por la que vemos la carretera desde una altura Z. Para proyectar la imagen en nuestra pantalla plana, supondremos que estamos mirando a través de una ventana a una distancia D.

Supongamos que tomamos un punto sobre la carretera a una

distancia Y1. Este punto aparecerá en la pantalla, debajo del horizonte una distancia SY. Habíamos supuesto antes que el horizonte se encuentra al nivel de la vista, como si estuviéramos mirando a lo largo de una línea paralela al eje Y. Por tanto, el pequeño triángulo entre la pantalla y el ojo tiene la misma forma que el triángulo grande que pasa por el punto Y1. Esto significa que todos los lados de los dos triángulos guardan las mismas proporciones, y, por tanto, podemos escribir:

SY/D = Z/Y1

que, reordenado, será

SY = Z * D/Y1

Observará que el tamaño de la imagen en la pantalla es inversamente proporcional a la distancia Y1. Si tomamos otro punto de la carretera a otra distancia Y2, producirá otra línea en pantalla de diferente longitud, siendo el nuevo valor de SY de:

SY = Z * D/Y2

Suponga ahora que tiene una serie de líneas de igual longitud dibujadas a lo largo de la línea central de la carretera. Cada línea producirá en pantalla una línea vertical corta, y, a medida que se vaya alejando el punto de la carretera, la imagen producida en pantalla por cada línea se acortará más aún.

Si dibujamos una vista general de la carretera, mirando ahora hacia abajo, encontraremos que la fórmula básica para el tamaño de la anchura de la carretera (SX) sobre la pantalla, es la misma. A medida que Y1 aumenta, la anchura de la imagen en la pantalla disminuye, de acuerdo con la fórmula:

SX = W * D/Y1

donde W es la anchura de la carretera.

Los objetos verticales a lo largo de la carretera también producen imágenes, que siguen la regla general de tener un tamaño inversamente proporcional a la distancia Y.

Comencemos con la carretera. En primer lugar, necesitamos dibujar la línea del horizonte, y lo haremos horizontalmente sobre la pantalla, ya que es nuestra referencia visual. Ahora, si dibujamos

líneas desde los dos ángulos inferiores hacia un punto situado hacia la mitad en la línea del horizonte, tendremos la carretera.

Supongamos que la carretera tiene 25 pies de ancho y que la estamos viendo en una pantalla de televisión de 14 pulgadas, a una distancia de unos 4 pies. La pantalla de televisión tendrá una anchura de un pie, en este caso igual a SX. D será 4, y X será 25. Tendremos, por tanto:

SX = D * Y/Y = 4*25/Y = 1

por tanto,

Y = 4 * 25 = 1ØØ pies

Vamos ahora a trabajar con el factor de escala en la dirección X de nuestra pantalla. Tenemos 256 unidades en el sentido de las X en la pantalla en el Spectrum, y para Y = 1ØØ la carretera llena la anchura de la pantalla, por tanto, SX deberá ser 256. Si escribimos la ecuación con un multiplicador, tendremos:

SX = SCX * X/Y

y, sustituyendo los valores que hemos calculado, tendremos lo siguiente:

256 = SCX * 25/100

extrayendo el término SCX, tendremos:

SCX = 256 * 100/25 = 1024

Para calcular los valores de SX, utilizaremos la ecuación:

SX = 1024 * X/Y

Apliquemos un proceso similar para calcular el factor de escala Y, suponiendo que nuestro punto de vista se encuentra a una altura de 10 pies, y que el punto se escribe en la parte inferior de la pantalla para una distancia de 100 pies. Supondremos que la línea del horizonte se encuentra en la pantalla en Y = 96. A partir de esto:

SY = SCY * 10/100 = 96

y

SCY = 960

por tanto, para calcular los puntos Y sobre la pantalla, utilizaremos:

SY = 960 * Z/Y

Para dibujar un punto sobre la misma carretera, haremos Z = 1Ø, y el valor de Y que utilizaremos en la instrucción de diseño será 96 — SY, ya que, al acercarnos, el punto se mueve hacia la parte inferior de la pantalla. Si tenemos un poste vertical, para dibujar la parte superior del poste restamos su altura de 10 para obtener el valor de Z. De este modo, un poste de 10 pies de altura producirá un valor Y de 96, y estará en línea con el horizonte, ya que estamos mirando desde una altura de 10 pies. Si el poste está más alto que diez pies, el valor de SY será negativo, y la parte superior del poste irá por encima de la línea del centro de la pantalla.

Para dibujar una vista en perspectiva de una carretera, con árboles, por ejemplo, necesitará establecer los valores de la altura de los troncos, y también la de las capas y la anchura de los árboles. Sabiendo la distancia de cada árbol al observador, podremos calcular sus coordenadas para la base, copa y puntos laterales, utilizando:

SX = 128 + 1024 * X/Y
SY = 96 — 960 * (1Ø—Z) Y

donde SX y SY son las coordenadas de dibujo de la pantalla, X es la distancia medida desde el centro de la carretera a la derecha (con valores positivos), y Z es la altura del objeto. Una vez conocidas las coordenadas, pueden dibujarse los árboles, y para ello utilizaremos los comandos PLOT y DRAW. Las marcas de la carretera se tratan de forma similar, excepto que, en este caso, Z será Ø.

La figura 9.15. muestra el listado de un programa para dibujar una vista en perspectiva de una carretera, y la figura 9.16 muestra el resultado en la pantalla.

```
100 REM Vista en perspectiva de una carretera.
105 REM Dibuja el cielo.
110 PAPER 5: CLS
115 REM Dibuja el desierto.
130 PAPER 6
140 FOR y=10 TO 21
150 FOR x=0 TO 31
160 PRINT AT y,x;" ";
170 NEXT x: NEXT y
175 REM Dibuja la carretera.
180 FOR x=-128 TO 127
190 PLOT INK 0;128,96
200 DRAW INK 0;x,-96
210 NEXT x
220 LET sx=1024: LET sy=960
225 REM Dibuja las marcas de la carretera.
230 FOR y=100 TO 2000 STEP 100
240 LET y1=INT (sy*10/(y+50))
250 LET y2=INT (sy*10/y)
260 LET x=INT (sx*1/(y+50))
270 FOR k=128-x TO 128+x
280 PLOT PAPER 7; INVERSE 1;k,96-y1
290 DRAW PAPER 7; INVERSE 1;0,y1-y2
300 NEXT k
310 LET x1=INT (sx*1/y)
320 FOR k=0 TO x1-x
330 PLOT PAPER 7; INVERSE 1;128-x,96-y1
340 DRAW PAPER 7; INVERSE 1;-k,y1-y2
350 PLOT PAPER 7; INVERSE 1;128+x,96-y1
360 DRAW PAPER 7; INVERSE 1;k,y1-y2
370 NEXT k: NEXT y: INVERSE 0
375 REM Dibuja el horizonte.
380 PLOT INK 0;0,96
390 DRAW INK 0;255,0
395 REM Dibuja los arboles.
400 LET tr=5: LET tt=20: LET tw=3
410 FOR y=150 TO 1200 STEP 150
420 LET y1=INT (sy*10/y)
430 LET y2=INT (sy*(10-tr)/y)
440 LET y3=INT (sy*(10-tt)/y)
450 LET x=INT (sx*12.5/y)
460 LET x1=INT (sx*tw/y)
470 LET x2=128-x: GO SUB 1000
480 LET x2=128+x: GO SUB 1000
490 NEXT y
500 STOP
990 REM Subrutina de dibujo de los arboles.
1000 PLOT INK 0;x2,96-y1
1010 DRAW INK 0;0,y2
1040 FOR k=-x1 TO x1
1050 PLOT INK 0;x2,96-y3
1060 DRAW INK 0;k,y3-y2
1070 NEXT k: OVER 0
1080 RETURN
```

Fig. 9.15. Programa que dibuja una carretera.

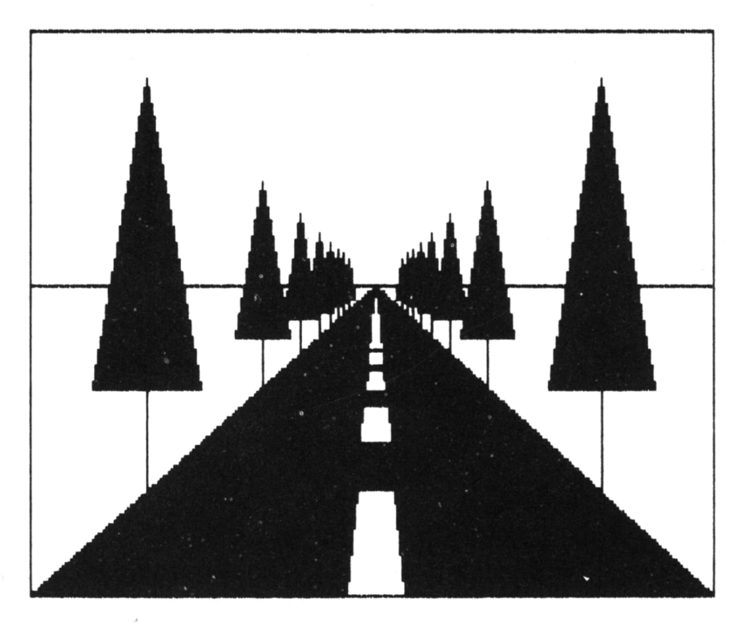


Fig. 9.16. Dibujo de una carretera.

Estructura de modelos de objetos con volumen

Si dibujamos imágenes en perspectiva de objetos tridimensionales, utilizaremos una matriz de coordenadas X,Y,Z para definir los puntos del objeto. Una vez dibujada la imagen, los puntos se unen por líneas para producir un reborde limitando el objeto. Si estamos dibujando un bloque rectangular, los puntos de referencia serán los ocho vértices del bloque, y las líneas que los unan representarán las aristas del bloque. Esta forma de dibujo se llama de IMAGEN ESTRUCTURAL, porque hemos creado el esqueleto de la figura marcando sus bordes.

Para mirar el objeto a lo largo de su eje Y, podemos utilizar la misma técnica de proporciones que se utilizó para la vista en perspectiva de una carretera. Si queremos ser capaces de movernos alrededor de nuestro objeto, es decir, verlo desde todas las direcciones, las ecuaciones son más complejas. También estudiaremos esta situación.

Visualización total de un objeto

Producir una vista en perspectiva de un objeto, desde cualquier punto alrededor de él, es un trabajo muy complicado. En primer

lugar, necesitamos definir el objeto como un conjunto de coordenadas X,Y y Z relativas a su centro. También necesitaremos información que relacione los bordes con los puntos X, Y, Z. Además necesitaremos datos para informar al ordenador cuándo debe trazar una línea y cuándo no, mientras está trazando el dibujo punto a punto en la pantalla. Estos problemas pueden solucionarse creando tres matrices de datos para describir el objeto. Las coordenadas X, Y y Z del objeto (incluidas en la matriz descriptiva) se miden con respecto al centro del objeto.

A continuación tomaremos un observador situado en un punto XV,YV, ZV con respecto al objeto.

La primera etapa del proceso es trasladar las coordenadas X, Y y Z del objeto, para que el espectador se sitúe en el origen de los ejes X, Y y Z. En este punto, las coordenadas X Y y Z son todas Ø. Este proceso de cambio es muy simple, ya que sólo implica sustraer XV, YV y ZV de X, Y y Z, respectivamente, para obtener un nuevo conjunto de valores de las coordenadas X, Y y Z de los puntos del objeto.

Hasta este momento, hemos supuesto que el espectador está a nivel. Para todas las perspectivas, podemos suponer que la línea de visión del espectador tiene tres componentes angulares, que llamaremos, respectivamente, rumbo, ángulo de picado e inclinación. Para entender todo esto, es útil imaginarse subido en un avión. El eje Y del avión se supone que está alineado con el fuselaje, y el eje X con las alas.

El rumbo es la dirección de la vista del observador, medida con respecto al eje Y. Un cambio en el rumbo equivale a torcer a la derecha o a la izquierda. Esto produce, de hecho, una rotación del avión sobre su eje vertical Z.

El siguiente término, ángulo de picado, indica si usted mira por encima o por debajo de la horizontal. Esto es, si el avión tiene la nariz hacia abajo o hacia arriba, y es equivalente a girar el aeroplano sobre la línea de las alas, es decir, una rotación sobre el eje X del avión. Finalmente, la inclinación indica si nuestra vista está girada hacia la derecha o hacia la izquierda, esto es, girar el plano alrededor de la línea de su fuselaje, que es lo que ocurre cuando gira su avión. En este caso, la rotación es alrededor del eje Y de éste.

Después del cambio del origen del mapa X,Y,Z, tendremos como posición el origen, y el objeto que miramos estará en el eje Y; pero el observador, de hecho, está mirando a un punto fuera del eje Y, debido a la desviación producida por considerar el ángulo que marca su rumbo. Con ello, el objeto se situaría fuera de la pantalla.

Para situar el objeto en el centro del campo visual y con una orientación correcta para los tres ángulos de rumbo, picado e inclinación, necesitamos girar todos sus puntos sobre los tres ejes. Esto se obtiene en tres etapas.

En la primera, los puntos giran alrededor del eje Z hasta que el observador mire directamente a través del centro del objeto. Se consigue esto rotando todos los puntos del objeto al rumbo del observador. Las ecuaciones de rotación son las mismas que utilizábamos en el capítulo 4, pero ahora están aplicadas a X, Y y Z. En su primera rotación, Z1 será igual al valor original de Z, ya que estamos girando alrededor del eje Z. Los nuevos valores de X e Y serán:

$$X1 = X * COS (H) - Y * SEN (H)$$
$$Y1 = X * SEN (H) + Y * COS (H)$$

En la segunda etapa de los cálculos, estos valores X1, Y1, Z1 se rotan el ángulo de picado, para producir un nuevo conjunto de valores X2, Y2 y Z2.

En esta rotación, los valores de X son invariables, ya que los puntos giran alrededor del eje X. El segundo conjunto de valores X2, Y2, Z2 soporta otra rotación, esta vez el ángulo de la inclinación, para producir las coordenadas X3, Y3, Z3. En esta rotación, producida en este caso alrededor del eje Y, los valores Y2 no cambian.

Una vez obtenidas estas coordenadas después de las rotaciones, habremos alcanzado aproximadamente la posición en la que estábamos mirando la carretera. Nos quedan por proyectar los puntos en la pantalla, y esto implica básicamente dividir X3 y Z3, respectivamente, por Y3, para obtener las coordenadas de pantalla XS e YS.

Todo este proceso de rotación y proyección se lleva a cabo para las coordenadas de cada punto del objeto, y entonces se trazan las líneas apropiadas entre los puntos, para formar el dibujo en la pantalla.

El programa listado en la figura 9.17. lleva a cabo este proceso en un objeto que es un modelo de estructura de un bloque. La cara delantera del bloque tiene una cruz dibujada para que resulte más sencillo interpretar el dibujo. Si introducimos la dirección de la visión, en términos de rumbo, ángulo de picado e inclinación, como si lo estuviéramos viendo desde la posición del observador, visualizaremos el objeto.

En efecto, estamos rotando el objeto mismo para presentarlo en la visión correcta. La figura 9.18. muestra un aspecto típico de todo ello sobre la pantalla.

```
100 REM Vision tridimensional de un hexaedro.
120 DIM v(50,3): DIM e(100): DIM l(100)
130 LET sx=10: LET sy=10
135 REM Establece los datos de la figura.
140 READ nv
150 FOR p=1 TO nv
160 READ v(p,1),v(p,2),v(p,3)
170 NEXT p
180 READ ne
190 FOR j=1 TO ne
200 READ e(j),l(j)
210 NEXT j
220 REM Establece la posicion del observador.
230 LET d=80: LET p=22
240 LET r=0: LET h=45
245 REM Bucle del programa principal.
250 CLS : PRINT "Rumbo= ";h
260 PRINT "Ang. picado= ";p
270 PRINT "Inclinacion= ";r
280 LET h=h*PI/180
290 LET p=p*PI/180
300 LET r=r*PI/180
305 REM Calcula los multiplicadores.
310 GO SUB 1000
320 LET xv=-d*cp*sh
330 LET yv=-d*cp*ch
340 LET zv=-d*sp
350 REM Proyecta la imagen en la pantalla.
360 LET x1=0: LET y1=0
370 FOR j=1 TO ne
380 LET n=e(j)
390 LET x=v(n,1): LET y=v(n,2)
400 LET z=v(n,3)
410 GO SUB 1200
415 REM Comprueba si debe dibujar la linea.
420 IF l(j)=0 THEN GO TO 450
425 REM Dibuja la linea.
430 PLOT x1,y1
440 DRAW xs-x1,ys-y1
445 REM Reubica la posicin del cursor.
450 LET x1=xs: LET y1=ys
460 NEXT j
480 INPUT "Otro punto de vista(s/n)";a$
490 IF a$<>"s" THEN STOP
500 INPUT "Rumbo(grad)=";h
510 INPUT "Picado(grad)=";p
520 INPUT "Inclinacion(grad)=";r
530 GO TO 250
540 STOP
1000 REM Factores multiplicadores.
1010 LET ch=COS h: LET sh=SIN h
1020 LET cp=COS p: LET sp=SIN p
1030 LET cr=COS r: LET sr=SIN r
1040 LET m1=ch*cr-sh*sp*sr
1050 LET m2=-sh*cr-ch*sp*sr
1060 LET m3=cp*sr
1070 LET m4=sh*cp
1080 LET m5=ch*cp
1090 LET m6=sp
1100 LET m7=ch*sr-sh*sp*cr
```

```
1110 LET m8=-sh*sr-ch*sp*cr
1120 LET m9=cp*cr
1130 RETURN
1190 REM
1200 REM Cambia la posicion del observador.
1210 LET x=x-xv: LET y=y-yv: LET z=z-zv
1220 REM Gira el punto de vista.
1230 LET x3=m1*x+m2*y+m3*z
1240 LET y3=m4*x+m5*y+m6*z
1250 LET z3=m7*x+m8*y+m9*z
1260 REM Calcula las posiciones de x e y
en la pantalla.
1270 LET xs=128+INT (sx*d*x3/y3)
1280 LET ys=80+INT (sy*d*z3/y3)
1290 RETURN
1990 REM
2000 REM Numero de puntos.
2010 DATA 12
2020 REM Coordenadas x,y,z.
2030 DATA 5,-3,4
2040 DATA -5,-3,4
2050 DATA -5,-3,-4
2060 DATA 5,-3,-4
2070 DATA 5,3,4
2080 DATA -5,3,4
2090 DATA -5,3,-4
2100 DATA 5,3,-4
2110 DATA 4,-3,3
2120 DATA -4,-3,-3
2130 DATA 4,-3,-3
2140 DATA -4,-3,3
2199 REM
2200 REM Numero de aristas.
2210 DATA 20
2220 REM Datos de aristas y lineas.
2230 DATA 1,0,2,1,3,1,4,1
2240 DATA 1,1,5,1,6,1,7,1
2250 DATA 8,1,5,1,2,0,6,1
2260 DATA 3,0,7,1,4,0,8,1
2270 DATA 9,0,10,1,11,0,12,1
```

Fig. 9.17. Programa que visualiza un bloque en tres dimensiones.

Usted mismo podrá producir una figura diferente, estableciendo una nueva matriz de datos. Las coordenadas x, y y z definen los vértices del objeto. Los datos del borde muestran la secuencia de coordenadas con las cuales se trazan las líneas. Se dan por pares, siendo la primera el número de la coordenada, y el segundo número es un 1, si se va a escribir la línea, o un Ø, si no es necesario. Recuerde cambiar los valores de los datos, especificando el número de puntos y de aristas.

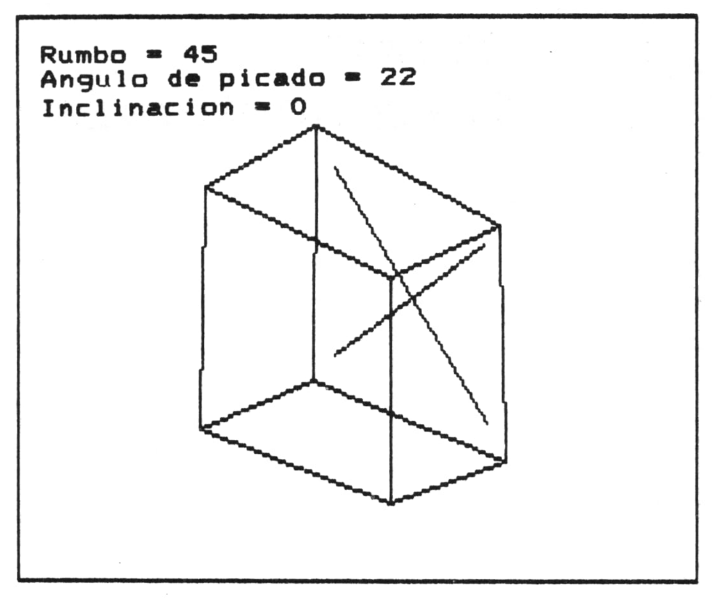


Fig. 9.18. Visualización típica del bloque.

Capítulo 10

Creación de música y sonido

Otro aspecto de los ordenadores personales que toma importancia en estos últimos años es la producción de sonido y de música. Esta faceta tiene su importancia cuando el ordenador se utiliza como máquina de juegos (usted se habrá fijado en éste último aspecto, examinando los video-juegos del mercado).

Algunos ordenadores personales modernos tienen sistemas de producción de sonido muy avanzados, con tres o cuatro canales de sonido independientes y control del volumen, frecuencia y duración de los sonidos que se producen. Este tipo de máquina es capaz de producir una variedad de sonidos casi infinita. La capacidad de sonido del Spectrum es mucho más modesta, ya que sólo tiene un canal y un comando BASIC para el control de la generación del sonido.

Como cualquier otro ordenador personal pequeño, el Spectrum tiene un altavoz incluido dentro de la carcasa del ordenador, para la producción del sonido. Inevitablemente, el altavoz es pequeño, ya que debe caber en el Spectrum, y este tipo de altavoces pequeños no da una buena reproducción del sonido. Otra limitación es que el sonido producido tiene muy poca intensidad. Sin embargo, puede obtenerse una señal de salida del conector MIC utilizado normalmente para el magnetófono cassette, cuando se almacenan los programas. Si la señal del conector Mic alimenta a un amplificador, se pueden obtener salidas con mucho mejor sonido, a partir del Spectrum. Existen varios amplificadores que pueden añadirse al Spectrum, y que utilizan la señal MIC para dar una salida de sonido más potente.

¿Qué es el sonido?

Todos los sonidos que escuchamos normalmente están producidos por ondas de presión que nos circundan por el aire. Para ver

cómo se produce el fenómeno, imagínese una piedra que cae al estanque. Cuando la piedra toca el agua, produce en la superficie una serie de ondas que se esparcen hacia fuera, partiendo del punto por el que entró la piedra. Las ondas sonoras son semejantes a estas ondas del agua, excepto que en el caso de las ondas sonoras éstas se deben a cambios en la presión del aire y son irradiadas a partir de la fuente de sonido. Cuando la onda sonora pasa por nosotros, el aire que está en contacto con nuestros oídos se comprime y se expande alternativamente, en simpatía con la onda sonora. En el interior del oído, las vibraciones producidas por la onda sonora se convierten en impulsos nerviosos, y notamos el sonido.

Un tono puro de sonido tendrá una onda de presión cuya forma sinusoidal de la onda determina la potencia del sonido. Normalmente nos referimos a ello como "volumen" del sonido. En el Spectrum no tenemos medios directos de controlar el volumen del sonido, y por tanto, todos los sonidos emitidos tienen el mismo volumen.

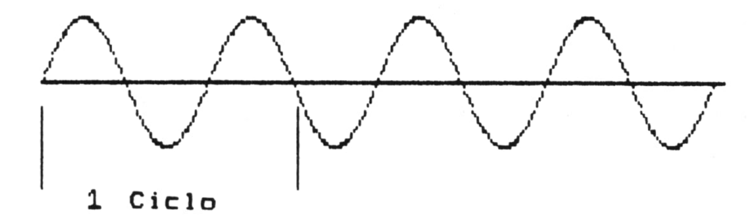


Fig. 10.1. Onda sinusoidal de sonido.

La rapidez con la que cambia la onda sinusoidal determina el tono o FRECUENCIA del sonido. Así pues, cuanto más deprisa cambie la onda, más alto será el tono. La frecuencia o tono de un sonido se mide como el número de ciclos completos de la onda sinusoidal que se lleva a cabo en un segundo; las unidades en las que se mide se llaman Hercios (Hz). En un ciclo, la señal de sonido comienza en cero. Sube hasta llegar a un máximo valor (positivo), vuelve a pasar por cero y llega hasta un máximo valor negativo, volviendo de nuevo a cero, como puede verse en la figura 1Ø.1. Así pues, una onda que tiene 5Ø ciclos por cada segundo, se dice que tiene una frecuencia de 5Ø Hz. Este sonido será un zumbido de tono bajo, y es el que se oye frecuentemente en los equipos eléctricos, ya que es la frecuencia de los principales accesorios eléctricos.

La gama típica de sonidos que puede detectarse por el oído humano, va desde 4Ø Hz hasta alrededor de 15.ØØØ Hz (15 kHz). En el Spectrum puede producirse una gama de frecuencias que va desde unos 1Ø Hz, en las bajas frecuencias, hasta unos 15.ØØØ por la parte alta.

Además de la frecuencia y el volumen, todos los sonidos tienen DURACION, que es la longitud de tiempo durante la cual se genera el sonido. El Spectrum permite controlar la duración de un sonido.

Una señal sonora de forma sinusoidal produce una nota pura similar a la que produce una flauta. Si cambiamos la forma de la onda, a una onda cuadrada, como la de la figura 10.2., se producirá un sonido más rico. La mayoría de los ordenadores tienden a producir este tipo de señales de sonido, ya que son mucho más fáciles de generar electrónicamente.

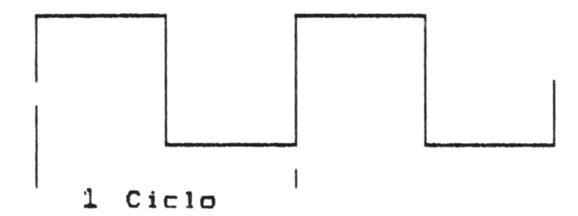


Fig. 10.2. Señal típica de una onda cuadrada.

Hasta ahora hemos considerado que la onda sonora tiene una frecuencia y un volumen constante durante toda su duración. En la vida real, la característica de un sonido está también muy influenciada por la forma en que varían la amplitud y la frecuencia, según se va produciendo el sonido. A esto se le conoce como ENVOLVENTE. Con el Spectrum no podemos variar la amplitud de los sonidos, y esto restringe los tipos de sonido que se pueden producir. Sin embargo, podemos variar la frecuencia con el tiempo para producir varias clases de efectos de sirena.

Generación de sonido con el Spectrum

En los ordenadores más sofisticados, se utilizan CHIPS (circuitos integrados) especiales generadores de sonido, que producen salidas controladas por el chip de la CPR del interior del ordenador.

Este esquema normalmente permite a la CPU del ordenador realizar otro tipo de tarea al mismo tiempo que se produce el sonido. En el Spectrum, el sonido se genera directamente con la misma CPU. La técnica básica es conectar una sencilla línea de salida y desconectarla a intervalos regulares, y utilizar la señal de esta línea para el altavoz. Cada vez que la señal se conecta y se desconecta, se produce un "clic" en el altavoz. Si producimos una secuencia de clics sucesivos, obtendremos un tono tosco, y la frecuencia del sonido que se genere dependerá de la rapidez con la que se produzcan los "clics".

La duración del tono producido con el Spectrum dependerá de la longitud de la sucesión de clics, y esto puede controlarse por el programa, como se hacía con la frecuencia. La mayor desventaja en esta técnica de producción de sonido es que, mientras el Spectrum está generando una salida de sonido, la CPU está ocupada totalmente por esta tarea, y el resto de la actividad del ordenador cesa. Evidentemente esto produce problemas, si usted está intentando producir gráficos animados al mismo tiempo que produce sonido. El problema se puede solucionar, sin embargo, rompiendo en pequeñas partes la señal de sonido, e intentando las modificaciones de la imagen con los comandos de sonido.

El comando BEEP

En el Spectrum, la instrucción de BASIC utilizada para la producción de sonidos se llama BEEP, y tiene el siguiente formato:

1ØØ BEEP duración, frecuencia

La duración del tono se mide en segundos, es decir, si la duración se ha fijado en 1, el tono durará un período de 1 segundo. Para ver cómo funciona, probemos a ejecutar el programita listado en la figura 1Ø.3. En la mayoría de los casos, el valor de la duración del sonido estará en una gama que va de Ø a 1.

```
100 REM Demostracion de la duracion del
sonido BEEP .
120 LET d=0.01
130 FOR n=1 TO 10
140 FOR n=1 TO 10
150 PRINT AT 1,1;"Duracion = ";d;" Segundos     ";
160 BEEP d,0
170 PAUSE 25
180 LET d=d*2
190 NEXT n
```

Fig. 10.3. Demostración de cambios en la duración del sonido.

El parámetro de la frecuencia es un número entero, y deberá estar en la gama que va desde —59 a +69. Un valor de —59 dará una frecuencia muy baja del zumbido, y, según va aumentando la frecuencia, aumenta el tono por toda la gama audible, pudiendo llegar a ser demasiado alta para oírse. En el nivel bajo de la frecuencia es unos 10 pulsos por segundo, y en el nivel más alto sube a unos 15.000 ciclos por segundo o 15 kHz (kilohercios). La nota Do, llamada en la escala del piano, que tiene una frecuencia de unos 261 ciclos por segundo, se produce con un número de frecuencia de P = Ø.

Para ver la gama de tonos disponible con el comando BEEP del Spectrum, intente ejecutar el programa que se muestra en la figura 10.4. En este programa, la frecuencia se modifica a intervalos por toda la gama de valores (—59 a 69) para dar una secuencia de tonos que van aumentando en frecuencia. La duración se ha fijado en 0.5 para dar tonos largos de medio segundo, y el comando PAUSE 12 se usa después de cada nota, para separar las notas individuales, con el fin de que puedan distinguirse fácilmente.

```
100 REM Gama de tonos del sonido.
110 FOR p=-60 TO 69
120 PRINT AT 1,1;"Tono = ";p;"     ";
130 BEEP 0.5,p
140 PAUSE 12
150 NEXT p
```

Fig. 10.4. Demostración de la gama de tonos disponible.

Los sonidos que usted puede producir añaden un interés considerable a la mayoría de los juegos de su Spectrum.

Creación de efectos sonoros

Con el comando BEEP, podemos producir algunos efectos sonoros sencillos. Si la frecuencia se ha programado para que suba y baje regularmente, como, por ejemplo, una sirena, podemos producirla con el programa listado en la figura 10.5.

Otras posibilidades de utilización del comando BEEP incluyen la obtención de dos sonidos en rápida sucesión, intercalando dos comandos BEEP en un bucle FOR.....NEXT, y con una pequeña diferencia entre ellos, tal y como se muestra en la figura 1Ø.6.

```
100 REM Sonido tipo sirena.
110 FOR p=5 TO 15
120 BEEP .05,p: NEXT p
130 FOR p=15 TO 5 STEP -1
140 BEEP .05,p: NEXT p
150 GO TO 110
```

Fig. 10.5. Producción de sonidos tipo sirena.

```
100 REM Notas intercaladas.
110 FOR p=1 TO 4
120 FOR n=1 TO 50
130 BEEP .05,5
140 BEEP .05,5+p
150 NEXT n
160 PAUSE 25
170 NEXT p
```

Fig. 10.6. Generación de una frecuencia tipo trino.

Este programa produce un tipo de sonido gorjeante mediante la sucesión de un trino de dos sonidos. Pruebe diferentes valores de la frecuencia del segundo sonido, para ver los efectos que produce.

Hagamos música

Como el Spectrum puede producir una amplia gama de frecuencias sonoras, puede también usarse para hacer música. Por supuesto, las limitaciones del sistema de generación del sonido no nos permitirán producir calidad en el sonido, pero, en cualquier caso, con el Spectrum pueden tocarse canciones. Todo ello puede ser útil en programas de juegos donde podrían sonar diferentes tipos de "campanas" cuando el jugador gana o cuando pierde.

Si consideramos una pieza de música, los sonidos que forman la canción se llaman generalmente NOTAS. Cada nota tiene una frecuencia determinada, relacionada con precisión con las otras notas de la escala musical. La gama completa de frecuencias utilizada en música, está dividida en grupos de notas que llamamos OCTAVAS.

Si examinamos una octava de notas, en un teclado de piano, por ejemplo, como el de al figura 10.7. vemos que consiste en siete notas llamadas NATURALES, que se producen con las teclas blancas del piano. Las notas naturales están etiquetadas, A, B, C, D, E, F y G (*) en orden de frecuencia ascendente. Esta secuencia de notas desde

* *N. del T. C = do, D = re, E = mi, F = fa, G = sol, A = la, B = sí.*

la A a la G se repite hacia arriba, de forma que la siguiente nota después de la G será la A que comienza la siguiente octava. Esta A es la octava nota natural sobre la anterior A del conjunto (de aquí es de donde deriva la palabra octava).

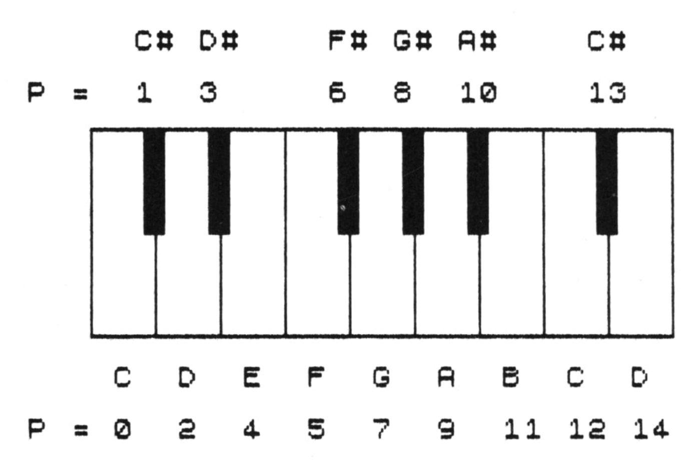


Fig. 10.7. El teclado del piano y los correspondientes números de las frecuencias.

Además de las notas naturales hay otras notas que se llaman CON SOSTENIDOS. Las notas con sostenidos son un semitono más altas (en frecuencia) que las notas naturales adyacentes. Las notas con sostenidos utilizan las mismas letras que las naturales, pero van seguidas de un símbolo. Así pues, la nota C# es un semitono más alta que la C.

También, en música encontrará referencia a notas CON BEMOLES, que son notas un semitono más bajas que las notas naturales. Los sostenidos y bemoles son verdaderamente formas diferentes de llamar a la misma nota. Si consideramos la nota A#, que es un semitono superior a la A, tendrá la misma frecuencia que la nota B bemol, que es un semitono inferior a B. En música, el bemol se escribe con un símbolo parecido a una b pequeña, que se coloca detrás de la nota.

Observará, por tanto, que en una octava sólo existen cinco notas con sostenidos, y siete notas naturales. La mayoría de las notas está separada de la siguiente nota natural por una nota con sostenido, por tanto, la diferencia entre las dos notas adyacentes naturales es dos semitonos o un solo tono. Las excepciones son B y E, que no tienen la nota sostenida correspondiente, y, por tanto, la diferencia entre B y C o entre E y F sólo es de un semitono. Esta ordenación

parece muy rara, pero funciona muy bien en la práctica, y, si tocamos las notas naturales una tras otra, tendremos la escala musical que nos es tan familiar.

Obtención de notas musicales

Puesto que la relación entre dos notas sucesivas es la duodécima raíz de dos, las frecuencias resultantes para las notas musicales son números bastante extraños. Por ejemplo, la nota Do (C) tiene una frecuencia de 261.7 Hz. Esta nota (C media) o Do central es la más frecuentemente utilizada como referencia en sistemas de sonido con ordenadores. El Spectrum no es una excepción a la regla. En el Spectrum, la frecuencia BEEP se especifica como un número de semitonos relacionado con el Do anterior, y, por tanto, si utilizamos el comando:

BEEP 1,Ø

la máquina producirá un DO (C media) por un período de un segundo. Un número de frecuencia positivo indicará el número de semitonos por encima de la nota Do (C), y un número negativo indicará el número de semitonos por debajo de la nota C (Do).

Para producir las notas familiares, "do re mi" de la escala, podemos tocar las notas naturales desde el Do (C) hacia arriba, e incluir la nota Do siguiente de la octava superior. En este caso, no podemos utilizar un bucle sencillo, como hicimos antes para demostrar la gama de los sonidos, ya que los números de las frecuencias que necesitamos no están igualmente espaciados.

La escala correcta puede tocarse estableciendo las frecuencias de las notas que deseamos en forma de matriz, y repitiendo el comando BEEP en un bucle en el que se varíe la frecuencia. Este programa se muestra listado en la figura 1Ø.8.

```
100 REM Programa de una escala musical.
110 FOR k=1 TO 20
120 RESTORE
130 FOR n=1 TO 8
140 READ p
150 BEEP .25,p
160 NEXT n
170 PAUSE 25
180 NEXT k
190 DATA 0,2,4,5,7,9,11,12
```

Fig. 10.8. Sonidos de la escala musical.

Las escalas pueden también tocarse comenzando desde otra nota diferente, pero, para obtener la secuencia correcta de sonidos, deberemos utilizar alguna de las notas sostenidas de la escala.

Interpretación de música escrita

Hasta ahora hemos producido una escala musical y podemos continuar tocando una canción, pero únicamente hemos escrito la secuencia de notas con sus nombres (las letras), y luego las hemos convertido en números de frecuencia, para el comando BEEP.

En la práctica, sin embargo, la música no se escribe en forma de secuencia de nombres de notas. Vamos ahora a estudiar cómo interpretar la música escrita.

La música escrita sobre papel utiliza notas que se escriben con forma de un punto grande con una cola vertical. Estos símbolos de notas se escriben sobre o entre una serie de líneas horizontales, cuyo conjunto se llama PENTAGRAMA, y que se muestra en la figura 10.9. La posición de la nota en el pentagrama indica su frecuencia, de forma que cuánto más alta esté situada en el pentagrama, más alta será su frecuencia. Las notas naturales sucesivas de la escala se escriben sobre y entre las líneas del pentagrama.

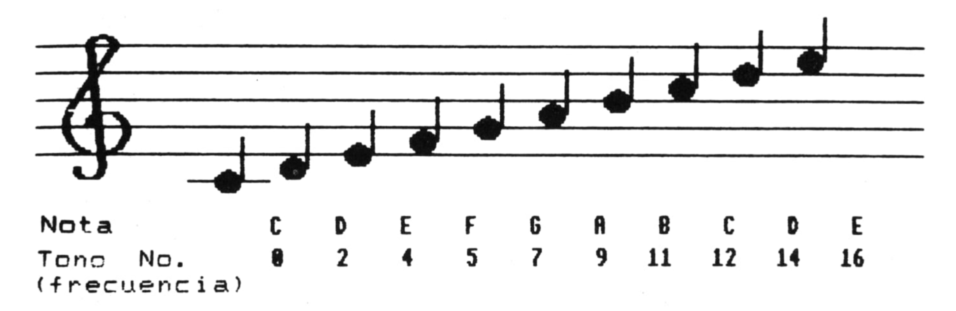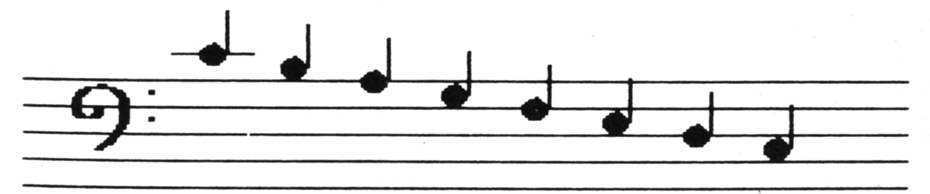


Fig. 10.9. Pentagrama musical con las notas y los valores de las frecuencias (tonos).

Existen dos claves musicales, una de ellas se muestra en la figura 1Ø.9., y se conoce como clave de Sol, identificando el conjunto de líneas como clave de sol. Esta clave muestra las notas que van a partir del Do (C media), que es la nota situada con una pequeña línea debajo del pentagrama.

Para las notas que están por debajo del Do, existe otra segunda clave, llamada CLAVE DE FA, y que se muestra en la figura 10.10. De nuevo tiene un signo especial a la izquierda, que es el que identi-

fica la clave. En la clave de Fa, la nota Do aparece con una pequeña línea por encima del pentagrama.



Nota	C	B	A	G	F	E	D	C
Tono No. (frecuencia)	0	−1	−3	−5	−7	−8	−10	−12

Fig. 10.10. Clave de Fa.

Las figuras 10.9. y 10.10. también muestran la relación existente entre las notas escritas musicalmente en las claves, y los números de las frecuencias que necesitamos utilizar en el comando BEEP, con el Spectrum.

Hasta ahora, sin embargo, hemos utilizado sólo las notas naturales, que corresponden a las teclas blancas del piano. Muchas piezas de música utilizan también las teclas negras o sostenidas, y, por tanto, debemos ser capaces de reconocerlas en la música escrita e informar al Spectrum para que las interprete.

Las notas con sostenidos se escriben musicalmente, colocando un signo especial detrás de la nota, como se muestra en la figura 10.11. En algunas canciones, una determinada nota deberá llevar un sostenido para que la canción suene bien, y para ello se colocará un signo de sostenido al comienzo de la línea de música en la posición ocupada normalmente por la nota. Esto también se muestra en la figura 10.11. Cuando el símbolo se coloca al comienzo del pentagrama, todas las notas de esa línea (y con ese nombre) se ven afectadas por esa alteración (sostenido), y ya no son naturales.

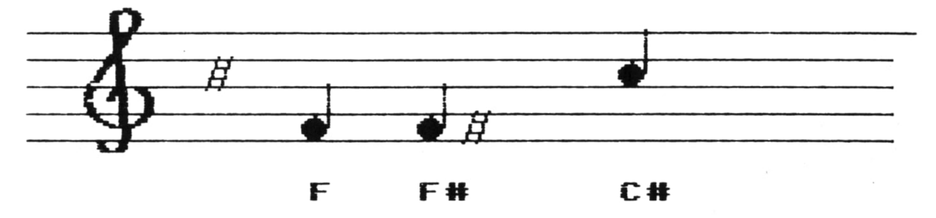


Fig. 10.11. Notas con sostenidos.

Es fácil informar al ordenador que debe tocar una nota con un sostenido, ya que sólo hará falta añadir uno al número de la nota básica. La nota sostenida (Do# o C#) tendrá un número de frecuencia de 1, ya que C (Do) tenía número de frecuencia Ø. En los casos en los que la nota se vea afectada con un bemol, bastará con sustraer una unidad al número básico de la frecuencia.

Ahora podemos producir una canción sencilla, que usted reconocerá, simplemente ejecutando el programa listado en la figura 10.12. Aunque las notas tienen todas ellas los valores de frecuencia adecuados, la canción no suena demasiado bien. Se debe a que todas las notas tienen la misma longitud, y en la música las notas tienen una duración variable, que es la que crea el ritmo de la canción.

```
100 REM Programa que interpreta una melodia.
110 FOR n=1 TO 26
120 READ p
130 BEEP .25,p
140 NEXT n
150 DATA 11,14,11,11,14,11,12
160 DATA 14,12,9,11,12,11,7
170 DATA 11,14,11,11,14,11
180 DATA 12,14,12,9,11,7
```

Fig. 10.12. Programa que interpreta una melodía sencilla.

Medida musical

Además de la frecuencia (tono), en música se utiliza la duración variable de las notas, y esto se organiza en un sistema binario. La longitud básica de una nota se llama NEGRA, y corresponde a una duración de 1/4 segundos. La negra tiene forma de círculo negro con una cola. Notas más cortas que la negra son la corchea, la semicorchea y la fusa (1/2, 1/4 y 1/8, respectivamente), que se escriben como la corchea, pero llevan uno, dos o tres ganchos en la cola. Notas más largas que la negra son la BLANCA, que vale doble que la negra y que se escribe igual que la negra, pero con el círculo sin llenar, y finalmente la redonda, que vale cuatro negras y no tiene cola. Los símbolos correspondientes a estas figuras se muestran en la figura 10.13.

En el Spectrum controlamos la longitud de la nota alterando la duración del comando BEEP. Una negra corresponde a un valor de 0.25, y las otras notas siguen la proporción que se muestra en la figura 1Ø.13. Esto nos obligará a utilizar series completas de números fraccionarios para el valor d.

Nota	Duración	Nombre	Longitud
(fusa)	1/8	Fusa	1
(semicorchea)	1/4	Semicorchea	2
(corchea)	1/2	Corchea	4
(negra)	1	Negra	8
(blanca)	2	Blanca	16
(redonda)	4	Redonda	32

Fig. 10.13. Longitudes de las notas.

Una solución para este problema puede ser fijar un valor básico de duración en 1/32 segundos, utilizando la variable c. Esto dará a la nota el valor de una fusa, que es la nota más corta. Podemos, a continuación, asignar a la longitud de una nota el valor l, que sea proporcional a la longitud de la nota. Con ello, la fusa tendrá valor 1, la negra valor 4, y la redonda un valor de 16. En la instrucción BEEP, la duración se obtiene multiplicando c por la longitud de la nota deseada l.

Ahora ya podemos aplicar estos valores de duración de las notas a la generación de nuestra canción, con el programa que se muestra en el listado de la figura 1Ø.14. En este programa, se utilizan dos matrices de datos, una de las cuales da las frecuencias (tonos), p, y la

```
100 REM Programa que interpreta una melodia
110 REM con notas de diferente duracion.
120 LET c=1/32
130 FOR n=1 TO 26
140 READ l,p
150 BEEP c*l,p
160 NEXT n
170 DATA 8,11,4,14,8,11,8,11
180 DATA 4,14,8,11,4,12,4,14
190 DATA 4,12,8,9,4,11,4,12
200 DATA 4,11,8,7,8,11,4,14
210 DATA 8,11,8,11,4,14,8,11
220 DATA 4,12,4,14,4,12,8,9
230 DATA 4,11,8,7
```

Fig. 10.14. Versión mejorada del programa de una melodía.

otra, la longitud, l. Los valores de p y l se usan ahora uno tras otro en la instrucción BEEP para tocar la canción, que ahora ya suena mucho mejor.

Las series binarias de notas no satisfacen, sin embargo, todas las necesidades musicales, y, en algunos casos, en la música escrita aparecen algunas notas con un punto junto al símbolo de la nota, tal y como puede verse en la figura 1Ø.15. Estas notas tienen su duración incrementada en la mitad de su valor. Así pues, una negra con puntillo tendra una duración correspondiente a un factor multiplicador 12 en lugar del normal, 8, y una corchea un puntillo tendrá un valor de 6 en lugar del normal, 4.

Otra característica de la música, aparte de la duración misma de las notas, es la existencia de pausas entre notas, que se conocen como SILENCIOS. Existen unos símbolos especiales para indicar estos silencios en el pentagrama, y se muestran en la figura 1Ø.16. Estos silencios pueden introducirse en el Spectrum, utilizando el comando PAUSE, pero para ello necesitaremos algún método que informe al Spectrum que deseamos PAUSE, y no una nota. Una posibilidad podía ser hacer negativa la longitud de la nota cuando deseemos un silencio, y, con una sencilla comprobación del número correspondiente a la duración, tendremos la indicación del tipo de instrucción que se requiere. La otra posibilidad es tener una tercera matriz para los silencios.

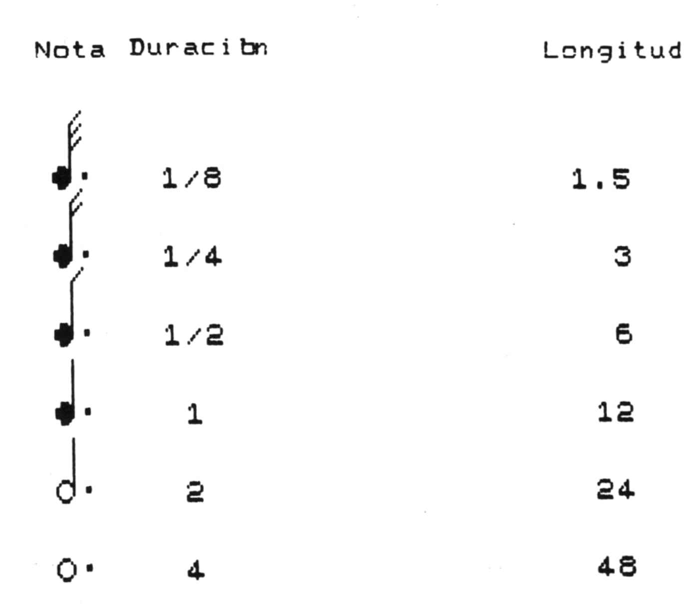


Fig. 10.15. Las notas y su duración.

Estos tendrán el valor 0 cuando se desea una nota, y el número que indica su duración, cuando se desee un silencio. Observe que puesto que el comando PAUSE trabaja con un tiempo de 1/50 segundos, las unidades de duración de los silencios pueden no ser exactamente correctas, pero bastarán para producir música aceptable.

Símbolo	Duración	Duración del Silencio
𝄿	1/16	1
𝄾	1/8	2
𝄽	1/4	4
𝄼	1/2	8
𝄻	1	16

Fig. 10.16. Símbolos musicales de los silencios.

Cambios de tiempo

Normalmente podemos tocar música fijando la longitud de una negra en 0.25 segundos. Si esta duración se hace inferior, la canción se ejecutará más deprisa, y si la longitud de la negra se aumenta, la canción se ejecuta más despacio.

Ejecute el programa listado en la figura 10.17. En él, se toca la escala básica desde el Do hasta una octava más arriba, incrementando a la vez el tiempo, hasta que, en un momento dado, todas las notas se amontonan y producen un sonido que podría ser útil para algún juego. Pruebe esta misma idea, pero bajando la escala.

```
100 REM Escala con cambios de   tiempo.
110 FOR t=0.25 TO 0 STEP -.02
120 RESTORE
130 FOR n=1 TO 8
140 READ p
150 BEEP t,p
160 NEXT n
170 PAUSE 25
180 NEXT t
190 DATA 0,2,4,5,7,9,11,12
```

Fig. 10.17. Aplicación de los cambios de tiempo a la escala para obtener efectos sonoros.

Un punto a tener en cuenta es que cuanto más corto sea el tiempo, más cortas se hacen algunas notas muy pequeñas, y algunas tienden a desaparecer. Esto se debe a que no hay tiempo para generar un ciclo de la nota deseada.

"Interpretar" el Spectrum

Hasta el momento hemos producido música fijando el conjunto de notas, y luego tocándolas una tras otra con el comando BEEP. Otra posibilidad mucho más atractiva con el Spectrum es convertirlo en un instrumento capaz de interpretar nuestra música.

Podemos utilizar el teclado del Spectrum para que actúe del mismo modo que un teclado de piano, introduciendo notas en unas cuantas teclas seleccionadas. En el momento en el que se pulse esa tecla determinada, sonará la nota correspondiente en el Spectrum.

El primer paso es elegir las teclas que se van a usar para producir las notas musicales. Es obvio que debemos intentar tener un teclado similar al del piano. Con esta idea se eligieron las teclas de las filas superiores de las letras. La fila intermedia de letras, que va de la A a la L, se utiliza para las notas naturales (las teclas blancas), comenzando por al tecla A que produce el Do natural. Algunas de las teclas de la fila superior, que va desde la Q a la P, se utilizan para las notas sostenidas (las teclas negras).

Para detectar qué tecla se ha tocado, podemos utilizar el comando INKEY$ que produce una cadena variable a$. Si no se toca ninguna tecla, la cadena a$ estará vacía, y esto último se comprueba en la línea 18Ø. Una cadena vacía hace que el ordenador repita la operación INKEY$. Una vez detectada la tecla, el siguiente paso es elegir y establecer el dato de la nota para el comando BEEP.

La técnica más sencilla para fijar las notas, es producir una matriz p con un valor para cada una de las teclas de letras. De hecho, como sabemos que no vamos a utilizar la Z, sólo habrá 25 posiciones en la matriz de datos. En cada posición se coloca un número del tono (frecuencia). Las teclas que van a utilizarse como notas, tienen un tono (frecuencia) que corresponde a la tecla que se va a simular.

El resto de las teclas utiliza un valor de 64 que produce un sonido muy alto y no puede oírse.

Los códigos de tecla para las letras desde la A a la Y comienzan con un valor de 97 para la A, y van aumentando según avanzamos por el alfabeto. En el comando BEEP, el código de la letra que se ha pulsado se puede obtener utilizando el comando CODE, y luego

restando 96 para obtener un número desde 1 a 25, dependiendo de la tecla presionada. Este número selecciona el término de la matriz p y, por tanto, la frecuencia correspondiente, para el comando BEEP. La longitud básica de las notas se ha fijado en Ø.25 para el comando BEEP, pero puede cambiarse si se desea, para darle otra sensibilidad al teclado.

La figura 10.18, lista un programa que convierte al Spectrum en un instrumento musical. Antes de que el programa pase al modo de instrumento, aparecerá un diagrama dibujado en pantalla para ayudarle a manejarse por el teclado. El dibujo que se verá en la pantalla es similar al mostrado en la figura 10.19.

```
100 REM Tocar con el Spectrum.
110 DIM p(25)
115 REM Establece la tabla de tonalidades.
120 FOR n=1 TO 25
130 READ p(n)
140 NEXT n
150 DATA 0,64,64,4,3,5,7,9,64,11,12,14,64
160 DATA 64,13,64,64,64,2,6,10,64,1,64,8
170 GO SUB 500
175 REM Bucle de notas.
180 LET a$=INKEY$: IF a$="" THEN GO TO 180
190 BEEP .25,p(CODE a$-96): GO TO 180
200 STOP
490 REM
495 REM Subrutina de dibujo de la presentacion.
500 LET b$=" "+CHR$ 133+CHR$ 138
510 FOR r=1 TO 4
520 PRINT AT 3+r,3;b$;
530 FOR k=1 TO 7
540 IF k=2 OR k=6 THEN PRINT "   ";: NEXT k
550 PRINT b$;: NEXT k
560 NEXT r
570 PRINT AT 2,5;"w  e     t  y  u     o";
580 PRINT AT 13,3;"a  s  d  f  g  h  j  k  l"
590 FOR k=0 TO 8
600 PLOT 16+k*24,80
610 DRAW 24,0
620 DRAW 0,64
630 DRAW -24,0
640 DRAW 0,-64
650 NEXT k
660 PRINT AT 17,1;"Ya puede tocar.";
670 RETURN
```

Fig. 10.18. Programa para utilizar el Spectrum como instrumento.

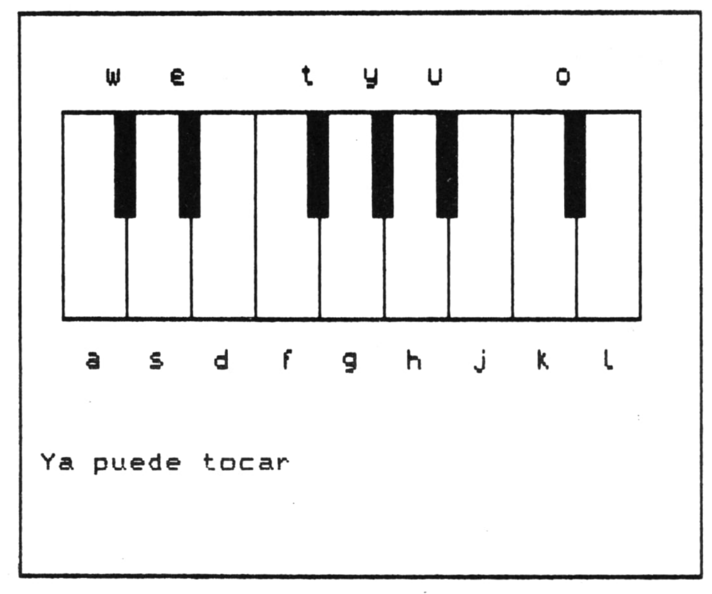


Fig. 10.19. Visualización en la pantalla empleada en el programa de utilización del Spectrum como instrumento.

Pueden utilizarse también otras teclas para obtener otras funciones, como, por ejemplo, cambiar la octava que se está utilizando. Para realizarlo, añada 12 al número del tono (frecuencia) para subir una octava, o reste 12 para bajar la octava.

Indice